Database Recovery Control (DBRC) in Practice

by Peter Armstrong

Third Edition

August 1998

bmcsoftware

Restricted Rights Legend

Contacting BMC Software

You can access the BMC Software Web site at **http://www.bmc.com**. From this Web site, you can obtain general information about the company, its products, special events, and career opportunities. For a complete list of all BMC Software offices and locations, go to **http://www.bmc.com/corporate/offices.html**.

USA and Canada

Address	BMC Software, Inc. 2101 CityWest Blvd. Houston TX 77042-2827
Telephone	713 918 8800 or 800 841 2031
Fax	713 918 8000

Outside USA and Canada

Telephone	(01) 713 918 8800
Fax	(01) 713 918 8000

Contents

Figures

Preface

Those who know me will know that I have spent the last few years advocating the fact that IMS is not dead. Fortunately, both customers and IBM* now agree with me! IMS is alive and kicking and not going away. Furthermore DBRC is a prerequisite to most of it—in fact, running a production IMS system without DBRC is equivalent to shooting yourself in both feet. The forced migration to DBCTL in the CICS environment and the move towards n-way sharing has also made many customers realize that they can no longer ignore my favorite piece of software!

Hence, people have been pestering me to update this document. For a long time I could not, as the source was in some strange piece of software running only on a Mac. Then, suddenly someone said that they could convert it all to a PC format for me—curses, my excuses fell away! So here is the updated version including the bits of IMS V6 I have played with. I have not included anything on RSR, as BMC Software has already produced a snappy little manual called *RSR in Practice*. One of the best things about V6 is that they have brought back the DBRC reference manual, rather than hiding it away in the depths of some other manual, where you can't find it.

I have assumed that you are running IMS V4 or above—previous versions are only mentioned when it helps to make things clearer.

Because this document is printed in America, I have had to grudgingly submit to their strange way of spelling—lovely people, just can't spell!

This manual is supplied on an "as-is" basis. I have tested the information to the best of my ability and believe the contents to be true. If you find DBRC behaves differently in your shop to the way described in this book, it could be due to various reasons such as maintenance levels, etc. If you are unable to determine why, contact me and we'll discuss it.

Finally, many thanks to all those people who have helped produce this, either through being friendly enough to call me with your problems, running tests for me, or being conned into proof-reading this set of ramblings.

Peter Armstrong—peter_armstrong@bmc.com
BMC Software, Inc.

Conventions

Because most of the examples in this book come from my work with BMC Software customers, you will find occasional refers to *customers*, their problems, and their solutions.

The term *IMS* refers to all versions and releases of IMS/VS and IMS/ESA. The specific product name, version, and release numbers are noted only when this information is significant.

Several special visual conventions are used in this manual to make it easier to use. They are as follows:

Note: Notes alert you to aspects of a discussion or instructions that deserve special attention.

Warning! Warnings alert you to situations that could cause loss of data or serious damage to your system if you do not follow the instructions carefully.

`All syntax and example JCL is presented in this typeface.`

In the text of a paragraph, commands are presented in THIS TYPEFACE.

Chapter 1 Database Recovery Control

Suffering from insomnia, I was once reduced to reading the Database Recovery Control (DBRC) reference manual from cover to cover. I discovered that while it is a good reference document, it certainly does not tell you how DBRC actually works.

The objective of this book, therefore, is to discuss how DBRC works in practice. Some of the topics discussed in this book include

- the principles of DBRC—why it was written
- how to implement DBRC
- how to use it
- problem areas to avoid
- techniques people have developed to make DBRC easier/possible

DBRC Background—Why do I need it?

DBRC was originally written to keep track of Information Management System (IMS) logs, change accumulations, and other log and tracking information. Armed with this information, DBRC can select the correct inputs and outputs for backing up and recovering databases, and for running change accumulation.

DBRC has been enhanced considerably since its early days. DBRC is now a prerequisite to many features including:

- Generating any IMS Transaction Manager (TM) or Database Control (DBCTL) system, or Customer Information Control System using Data Language 1 (CICS-DL/1) system.

  You cannot generate a system running IMS TM, DBCTL, or CICS-DL/1 without DBRC. DBRC is also required for maintaining these systems.

- IMS Disk Logging (online IMS)

  Whereas use of DBRC is optional in CICS-DL/1 or batch DL/1, you cannot run an IMS TM or DBCTL system without it. IMS TM uses DBRC to control the disk logging process—i.e. selection of the next online log, automatic archiving, etc.

  Note: As from IMS V3, DBRC is a prerequisite to the new database control (DBCTL) address space. This means that DBRC Log Control (next online log selection, archiving process, etc.) will be available to any system using DBCTL—for example, CICS.

  Note: IMS V4 is the last release to support CICS-DL/1. In the future, you have to run with DBCTL.

  In CICS-DL/1 systems and batch systems, DBRC can record logs. However, there is *no* automatic archiving and *no* control over the following items:

  — Selecting logs

    CICS selects journal A, B, or X as relevant. DBRC is just a recording mechanism here. For batch processing, log selection is dictated by your job control language (JCL).

- Reusing the same log prior to archiving it

  Exits in CICS enable you to trigger archiving. IBM manual GA19-5478 contains sample code for using them. Archiving is a user responsibility in batch systems. You must use generation data groups (GDGs), retention periods, etc., to control your batch logs.

  CICS V3.1 provides a degree of automation for the archive process. A far more complete solution is provided by the BMC Software JOURNAL MANAGER PLUS product.

- Share Control

  Sharing databases across one or two MVS images was introduced in IMS V1.2 for DL/I and in IMS V1.3 for Fast Path data entry databases (DEDBs). As from IMS V5, sharing has been extended to n-way sharing. This means that multiple IMS systems (up to 32) can now share data across multiple MVS images (up to 64). DBRC Share Control is required, and the following levels of sharing are allowed:

  - SHARELVL 0—enforced single user
  - SHARELVL 1—multiple readers or one update plus read-only
  - SHARELVL 2—multiple updaters in one MVS image
  - SHARELVL 3—multiple updaters across two or more MVS images

  Support for shared sequential dependents (SDEPs) and shared VSO DEDBs (see below) comes in IMS V6. General sequential access method (GSAM) and main storage databases (MSDBs) cannot be shared.

- Multiple Area Data Sets (MADS) in Fast Path

  Fast Path DEDB users can choose to maintain multiple copies of the data sets making up their DEDBs. This is known as maintaining multiple area data sets. DBRC Recovery Control or Share Control is required.

- Extended Recovery Facility (XRF)

  DBRC Share Control is required.

- IMS V3

  DBRC is a prerequisite for several features in IMS V3: for example, High Speed Sequential Processing (HSSP) and NONRECOV (non-recoverable databases).

- Remote Site Recovery (RSR)

  Refer to the BMC Software manual *RSR in Practice.*

- Virtual Storage Option (VSO) DEDBs

  IBM is gradually replacing main storage databases (MSDBs). The recommendation is to convert your existing MSDBs to VSO DEDBs (a VSO DEDB with only root segments is effectively the same as an MSDB, is supported by DBRC, can be block-level shared and also supports field calls). MSDBs are not supported by DBRC and neither is GSAM.

DBRC Levels

There are three levels to DBRC: Log Control, Recovery Control, and Share Control. In IMS V6, Recovery Control dies (about time too, frankly—if you're still using Recovery Control, then you are a sad individual and you should get yourself onto proper DBRC Share Control as soon as possible). I have come across several scenarios that work fine with Share Control, but do not work properly with Recovery Control.

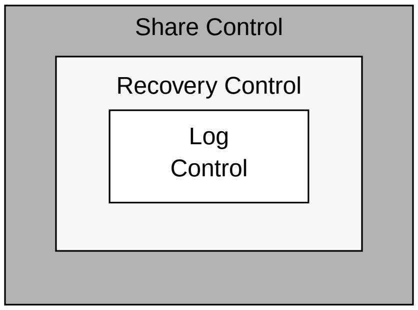


Figure 1 Levels of DBRC

Log Control

In Log Control, DBRC records information about logs. For CICS-DL/1, DBRC is optional. Turning on DBRC means that DBRC records the journals. If using DFSUARC0 (the recommended IMS archive utility) to archive the journals, the output is recorded in DBRC's control data sets—the RECONs.

For DBCTL and online IMS, DBRC is mandatory. It is installed as part of the IMS generation process (through the DBRC parameter on the IMSCTRL macro).

Follow these steps to implement Log Control:

Step 1 Initialize the RECONs with RECOVCTL (SHARECTL if DBCTL or V6 user).

Step 2 Do not register any databases with DBRC.

Recovery Control

In Recovery Control, DBRC provides all the Log Control functions and records information about image copies (IC), change accumulations (CA), recoveries, and reorganizations. To use Recovery Control, register your databases with DBRC.

Also, in Recovery Control DBRC generates and verifies JCL for image copy, change accumulation, and recovery. In addition, DBRC also records output from image copy, change accumulation, recovery, and backout. This means DBRC will avoid errors like missing logs, wrong image copy data set, recovery to an invalid point in time etc.

DSLOG is a DBRC unique utility that is *not* recommended, because support for it was discontinued in IMS V3.1. This book does not cover DSLOG.

Note: There is no GENJCL.BACKOUT. See Chapter 8, “Batch Backout, Signon, and Authorization” for a description of the backout process.

Warning! Recovery Control does *not* enforce the correct utility to be run at the correct time. For example, the user must control

- image copy after reorganization
- recovery after database I/O error
- backout after batch failure
- /ERE after online failure

Follow these steps to implement Recovery Control:

Step 1 Initialize RECONs with RECOVCTL.

Step 2 Ensure that DBRC is turned on in online systems, batch jobs, and utilities.

Step 3 Register your databases.

Share Control

Share Control contains all of the features of Recovery Control, plus it guarantees the following:

- Only valid sharing combinations run concurrently.
- The correct job/utility is run at the right time (see examples above), ensuring database integrity.

Follow these steps to use Share Control:

Step 1 Initialize RECONs with SHARECTL.

(If you are changing from Recovery Control, wait for all jobs running against the RECONs to complete, and then use CHANGE.RECON SHARECTL).

Step 2 Ensure that DBRC is turned on in online systems, batch jobs, and utilities.

Step 3 Register your databases.

Share Control is in fact required for DBCTL, but you do not have to register any of your databases. However, you are strongly recommended to do so—you should think of DBRC as a security mechanism like RACF. It should be turned on everywhere, you should not be able to turn it off, and everything should be under its control.

Recommended Migration Path

These are the recommended steps to follow for a new DBRC user:

Step 1 Play with all of this in a test environment and get familiar with it before you start in earnest.

Step 2 Implement DBRC Log Control.

Step 3 Turn on DBRC everywhere, and prove that all your jobs run with DBRC. You turn on DBRC via parameters in the IMSCTRL macro and/or overrides in your JCL. The most likely problems here are:

- S80A abend—DBRC requires 500K or more extra.

- No log—any job with a PROCOPT > G needs a log. Some customers run batch jobs without logging, e.g. image copy, run a suite of batch jobs, image copy again. While this is operationally simple (just recover to last image copy and rerun the jobs if anything goes wrong), it is inefficient and not recognized as valid by DBRC. To run this scenario, you would have to turn on DBRC for the image copies and turn DBRC off for the batch jobs.

Warning! Any implementation which involves turning DBRC on and off is *inherently dangerous.* FORCEing DBRC to be turned on (DBRC=FORCE on the IMSCTRL macro), and FORCEing registration of all your databases (FORCER parameter on INIT.RECON / CHANGE.RECON commands) should be your ultimate objective, as this combined with Share Control provides the only safe environment.

The BMC Software BATCH CONTROL FACILITY product contains a feature called Exchange Log Device Type, which enables you to implement logging in batch jobs and run them under DBRC control with *no* JCL changes. It will also allow you to run batch jobs with no log under DBRC control. While I don’t recommend the latter as a long-term solution to your batch operations, it is a great migration and testing environment aid.

- Wrong RECONs—this should not happen if you use dynamic allocation (recommended) and everybody runs from the correct RESLIB

Step 4 Change to SHARECTL. Prove everything works—for example, backout, recovery from a central processing unit (CPU) or power failure, etc.

Step 5 Register databases.

How It Works

The essential elements for operating DBRC are exits and hooks, RECONs, skeletal JCL, and commands.

Exits/Hooks

A log switch, job initialization, and other various parts of IMS code automatically invoke DBRC. When invoked, DBRC records and checks the events against its control data sets—the RECONs.

RECONs

DBRC keeps its information in a pair of virtual storage access method (VSAM) key-sequenced data sets (KSDSs). It is also recommended that you define an available empty RECON, which acts as a SPARE. See Chapter 2, “RECON Setup” for recommended options for setting up RECONs.

Skeletal JCL

DBRC generates JCL by taking a skeleton member from the JCLPDS DD library and replacing the DBRC keywords in the JCL with information from RECON. The installation process creates skeleton members that you can update to include buffering, accounting information, user IDs, passwords, etc. Full details of these members are given in the IBM manual *DBRC Guide and Reference.*

```
//AR%STPNO EXEC PGM=DFSUARC0,PARM='%SSID'
//*                 JCL FOR ARCHIVE UTILITY
//STEPLIB  DD  DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
%DELETE    (%RCNDSN1 EQ '')
//RECON1   DD  DSN=%RCNDSN1,DISP=SHR
%ENDDEL
%DELETE    (%RCNDSN2 EQ '')
//RECON2   DD  DSN=%RCNDSN2,DISP=SHR
%ENDDEL
%DELETE    (%RCNDSN3 EQ '')
//RECON3   DD  DSN=%RCNDSN3,DISP=SHR
%ENDDEL
%SELECT    OLDS(%SSID,(%ddnames))
//%OLDSDDN DD  DSN=%OLDSDSN,DISP=SHR
%ENDSEL
//DFSSLOGP DD  DSN=IMS.SLDSP.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             UNIT=3400,VOL=(,,,99),
//             DISP=(NEW,KEEP),LABEL=(1,SL)
//DFSSLOGS DD  DSN=IMS.SLDSS.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             UNIT=3400,VOL=(,,,99),
//             DISP=(NEW,KEEP),LABEL=(1,SL)
//RLDSDD1  DD  DSN=IMS.RLDSP.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             UNIT=3400,VOL=(,,,99),
//             DISP=(NEW,KEEP),LABEL=(1,SL)
//RLDSDD2  DD  DSN=IMS.RLDSS.%SSID.D%ARDATE.T%ARTIME.V%ARVERS,
//             UNIT=3400,VOL=(,,,99),
//             DISP=(NEW,KEEP),LABEL=(1,SL)
%ENDDEL
//SYSIN    DD  *
SLDS FEOV(08000)
COPY DDNOUT1(RLDSDD1) DDNOUT2(RLDSDD2) DBRECOV
/*
```

Figure 2 Sample Archive Skeletal JCL Member

Use the JCLOUT data definition (DD) statement to route the generated JCL to an internal reader (recommended) or a data set for browsing, editing, etc. For example:

```
//DBRC     EXEC PGM=DSPURX00
//STEPLIB  DD   DSN=IMSVS.RESLIB,DISP=SHR
//JCLPDS   DD   DSN=IMSVS.PROCLIB,DISP=SHR
//JCLOUT   DD   SYSOUT=(A,INTRDR)
//IMS      DD   DSN=IMSVS.DBDLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=A
//SYSIN    DD   *
  GENJCL.IC DBD(DI21PART) DDN(DI21PARO)
//
```

Figure 3 Sample DBRC Job Step

DBRC writes its SYSPRINT data set with a block size of 133. This does not matter if the file is allocated to SYSOUT, but if you are writing it to DASD for perusal by a program, this is very bad. Adding DCB=BLKSIZE=23408 to the allocation for this data set at one customer site reduced EXCPs from 75,425 to 416, and reduced the run time of the step from five minutes to a few seconds.

Note: You have to include the appropriate %SELECT commands, %DELETE commands, etc. in your skeletal JCL. These commands allow you to control under which conditions JCL is to be generated. Examples are given throughout this manual and IBM manual GG24-3333; they are explained in the *DBRC Guide and Reference* manual. There is also an excellent BMC Software publication I put together after some lengthy testing of this feature, which is rather boringly entitled *GENJCL.USER Examples and Description*—a riveting read designed to cure your insomnia problems.

Commands

You can enter DBRC commands in lots of different ways. Some customers use ISPF front-ends to make command entry easier—I personally think these are a waste of time as your objective in life should be to automate the interface to DBRC and not spend your life entering commands with 12-digit timestamps.

Try to set up automated procedures, using GENJCL.USER (see below), MVS procedures, IMS automated operator programs (available to DBCTL as from IMS V5), etc. Any production procedure that requires you to look in RECON or enter a DBRC command is a non-starter in my opinion. Do not use TSO foreground for DBRC, as the enqueues can cause other systems (e.g. online IMS) to hang waiting for your TSO session to finish.

It is not possible to execute the DBRC Recovery Control utility under the control of UCF (Utility Control Facility). No checking is done to prevent this, and the results are unpredictable (actually, they are totally predictable—it doesn't work properly!).

GENJCL

There are several useful parameters available on the GENJCL commands.

- **DEFAULTS**(member)

  is an optional parameter you use to specify the skeletal JCL default members to be used when generating JCL. Up to 10 members can be specified. Default members are searched to resolve keywords in the order in which the members are specified on this parameter.

  If a keyword is assigned a value in both the DEFAULTS and USERKEYS parameters, the value specified in USERKEYS is used.

- **JCLOUT**(JCLOUT/ddname)

  is an optional parameter you use to specify the output data set for the generated JCL. The data set is specified by ddname. A JCL DD statement with this ddname must be included in the job step containing the GENJCL command. The specified data set can be a member of a partitioned data set (PDS) as long as it is not the same data set used for the default JCLOUT.

  JCLOUT(JCLOUT) is the default.

- **JCLPDS**(ddname)

  is an optional parameter you use to specify the skeletal JCL data set to be used for input when generating JCL. The data set is specified by ddname. A JCL DD statement with this ddname must be included in the job step containing the GENJCL command. JCLPDS(JCLPDS) is the default.

- **JOB**/**JOB**(member)/**NOJOB**

  are mutually exclusive, optional parameters you use to specify whether to produce the job statement in the generated JCL.

JOB specifies that the job statement is produced. When JOB is specified without a member name, the IBM-supplied execution member JOBJCL produces the job statement, so if you are going to use this make sure that the JOBJCL member contains valid JCL for your environment. When JOB(member) is specified, the specified execution member produces the job statement. NOJOB specifies that the job statement is not produced in the generated JCL. JOB is the default.

- **LIST/NOLIST**

  are mutually exclusive, optional parameters you use to specify whether to print the generated JCL using the SYSPRINT data set.

  LIST prints the generated JCL. NOLIST suppresses printing of the generated JCL. NOLIST is the default. I use this a lot for testing—I generate JCL and list it to check it. When I've got it sorted out, then I route it to an internal reader to execute.

- **USERKEYS**(%key1,'value'/%key2)

  is an optional parameter you use to set the value of keywords you have defined. Up to 32 keywords can be specified.

  %key1 is the user-defined keyword being assigned a value. The maximum length of the keyword is 8 characters, including the percent sign. The first character after the percent sign must be alphabetic (A-Z). The remaining characters must be alphanumeric (A-Z, 0-9).

  value is the value assigned to the user-defined keyword when it is encountered. *value* can be any character string enclosed in quotes. The maximum length of *value* is 132 characters. If *value* contains a quote, use two single quotes. *value* can be a null string ('').

  %key2 is any simple keyword previously assigned a value, including DBRC and user-defined keywords.

Have a look in my *GENJCL.USER Examples and Description* manual for some scintillating examples of how to use these.

GENJCL and BMC Software

When you use GENJCL with the normal IMS utilities, it goes to RECON to extract the information required, combines this with your skeletal JCL and produces a job to execute. When this job starts, the first thing it does is to check that the JCL is correct by reading the RECONs again.

If you are using BMC Software utilities—IMAGE COPY PLUS, CHANGE ACCUMULATION PLUS, and RECOVERY PLUS—then you are recommended not to use GENJCL. Simply execute the BMC Software job for the appropriate group of DBDSs. The BMC Software job will then go to the RECONs, determine the relevant data sets, dynamically allocate these (if you have chosen that option—also recommended), and execute. This means only one read of the RECONs, instead of two.

Commands

The RECONs can be amended, interrogated, backed up, etc., using DBRC commands (such as LIST.RECON STATUS). These are all documented in the IMS manuals.

One common complaint is that there is no security mechanism for DBRC commands. Users are either allowed to do everything or nothing. What this means is that you have to give users Resource Access Control Facility (RACF) control authority, even if they only want to issue LIST commands.

PGM(DSPURX00) always opens RECONs for update, even if you are producing a listing.

IBM has recently announced a product called the IMS DBRC Security Tool to address these areas—I'm afraid I have not had time to play with it yet.

Chapter 2 RECON Setup

This chapter is a collection of pointers to various manuals about the best way to set up the RECON data sets.

The placement of the RECON data sets in the direct access storage device (DASD) configuration is very important. While the primary rule is to configure for availability, the performance implications should not be neglected. It is bad news to lose both of your active RECON data sets at the same time (to paraphrase Oscar Wilde, the loss of one RECON might be considered unfortunate, but the loss of both—or all three—is careless in the extreme).

See the *DBRC Guide and Reference* manual for further details on placement and a discussion of Global Resource Serialization (GRS) with DBRC.

Creating a RECON Data Set

A RECON data set is a VSAM KSDS. It is created by the use of a DEFINE CLUSTER command. The high-level identifier should be different for each RECON data set. This allows them to be cataloged in different catalogs (this restriction is lifted in DFP version 3). It is recommended the low-level identifier be RECON1, RECON2, or RECON3 respectively. However, anything that conforms to your company standards is acceptable.

You should try to keep the RECONs small, as they can become a performance bottleneck (especially when shared amongst multiple MVS images). Each job step accessing the RECONs uses ENQ/DEQ to “hold” the RECONs while doing its updates, so don’t submit 500 image copy jobs (one for each DBDS) at the same time—you will cripple your performance. Set up your jobs to copy and recover groups of database data sets in one job step.

Here are some things to keep in mind:

- Keep the number of image copy generations down to a sensible number. DBRC retains all recovery related records (IC, CA, ALLOC, LOGALL, RECOV, REORG, PRILOG, etc.) back to the oldest image copy in RECON. If you image copy once a week and keep 10 generations of image copy, then your RECONs will be at least 10 weeks big. Ask yourself how often you have had to go back to a previous generation of image copy. Three or four generations should be quite sufficient for most users. See also “GENMAX Cycle—Log Cleanup in RECON” on page 45 for further details.

- Keep the number of change accumulation generations down to a sensible number.

- Delete the INACTIVE log records/entries from RECON regularly—see “DELETE.LOG INACTIVE” on page 57.

- Reorganize the RECONs regularly—see “Reorganization of RECONs” on page 23.

- Consider compressing them with software like the BMC Software DATA ACCELERATOR Compression product.

- Put them behind a cache.

Here are some guidelines that one customer provided for allocating a primary space allocation for the RECON data set:

- The RECON data set holds approximately 30 days’ worth of data.
- Jobs causing the entries in the RECON data set are run constantly.
- They use 8K control intervals with spanned records and a maximum record size of 64K.
- They always use dual image copies.

The records contained in the RECON data set are as follows:

- 2 change accumulation group (CAGRP) records
- 15 change accumulation records per CAGRP record
- 286 database records
- 396 DBDS records
- 5 IC records per DBDS record with an average of one volume each
- 30 ALLOC records per DBDS record
- 1970 primary recovery log (PRILOG) records—mainly with one volume each
- 1970 secondary recovery log (SECLOG) records—mainly with one volume each
- 1970 LOGALL records

- 2 SUBSYS records
- approximately 100 RECOV/REORG records

This information utilizes eight cylinders of a 3380 disk. If your RECON starts growing beyond 20 cylinders, you are probably not cleaning it up properly.

The RECORDSIZE(avg,max) average parameter for a RECON data set should set to be 128. The maximum record size should be large enough to contain the largest recovery or system log data set anticipated. Should this be larger than 32,760 bytes, which happens to be the largest CI size allowed in VSAM, then use the SPANNED parameter to spread the record over several CIs. (In fact, if your RECORDSIZE for RECON is greater than 8K, you are recommended to use 8K CIs and specify SPANNED.)

There are several record types in the RECON, which can grow to larger than 32K. The most common problem is the PRILOG record—see “Size of PRISLD/PRILOG Record” on page 61 for further details. The other record that can grow if you have a large number of databases authorized to a subsystem is the SUBSYS record—the size of this is 72 + (number of authorized DBs x 32 bytes).

The key size changes from 24 bytes to 32 bytes when you go to V6—see below for UPGRADE procedures.

Use different sizes and different FREESPACE specifications for each RECON, and make all three RECONs different sizes so they don’t all fill up at the same time. Don’t change any of the VSAM parameters while the RECONs are in use. Wait until the active subsystems have terminated, then go through a procedure of backup, change, check, backup, and start again.

Ensure that the smallest data CI size specified exceeds the largest index CI size specified by at least 2048 bytes. Failure to do so can seriously degrade your DBRC performance.

RECON Buffering

It is not easy to find this in the manuals, but it makes a big difference to your RECON performance. Use BookManager and search on DSPBUFFS—you will then find the details you require in the *Customization Guide.*

Use CSECT DSPBUFFS, which is link-edited into DSPCINT0, to specify the number of buffers you want to use. If you can’t find this CSECT in your source libraries, it is due to the fact that it is supplied as part of the optional source (SVOPTSRC), which you have to pay extra for. Do not panic, simply use BookManager to find it (or wait for this book to come out on CD-ROM) and then cut and paste it across; or, of course, you could just type it in anyway!

```
    DSPBUFFS CSECT ....
    ....
LSRONLIN DC   AL2(6,12)    XA ENVIRON - IMS ONLINE DBRC
LSRCICS  DC   AL2(6,12)    XA ENVIRON - CICS USE OF DBRC
LSRBATCH DC   AL2(6,12)    XA ENVIRON - OFFLINE/BATCH DBRC
    ....
NSRONLIN DC   AL2(2,2)     NONXA ENVIRON - IMS ONLINE DBRC
NSRCICS  DC   AL2(2,2)     NONXA ENVIRON - CICS USE OF DBRC
NSRBATCH DC   AL2(2,2)     NONXA ENVIRON - OFFLINE/BATCH DBRC
```

IMS TM and DBCTL users should specify ten times the default number of buffers, as the buffers are allocated in the DBRC separate address space, where you have plenty of space. CICS-DL/1 (and batch) users should allocate as many buffers as their virtual storage permits.

At physical open time, the maximum buffers used are the LSRPOOL plus three times the NSR (i.e., 6 data + 6 index) = 168K (for 2K index and 8K data). In steady state, the maximum buffers used are the LSRPOOL plus two times the maximum RECORDSIZE.

Here are some further tips I picked up from a customer via the IMS list server:

Besides looking at the DSPBUFFS CSECT, you might want to try looking closely at a LISTCAT for the RECONS. If you're on 3390s, then the 16K CI size results in a 4K physical record size. Try going to 18K CI for the data component; you'll get an 18K physical record size resulting in just 3 records per track instead of 12. That's 1/4th the I/O for each block. Going to 18K also increases the DASD usage from 86.74% to 97.59%.

As far as recommendations for DSPBUFFS goes, try to keep an entire control area of data in memory. For the 18K CI, that's 45. If you let the index CI default, VSAM will pick 512K since there are less CIs/CA. Look at the LISTCAT to see how many index CIs exist, then pick a number that will keep a manageable amount in storage.

But don't forget that more buffers means more region is needed. If you increase the batch numbers too much, jobs will start running out of memory. The online numbers should be safe to increase since you control the region size for the IMS tasks.

Changing Versions—the UPGRADE command

When you migrate from one IMS version to another, the RECONs change (new record types, new record formats etc.), and you must apply an SPE to your previous system and UPGRADE the RECONs to the new level. The UPGRADE command comes as part of a Migration/Coexistence SPE for your current IMS version and uses a special DBRC utility—DSPURU00 (also part of the SPE), not the usual DSPURX00.

Prior to V6, this UPGRADE followed the following steps:

- Back up both RECON data sets.
- Install the Migration and Coexistence SPE.
- Back up.
- Install the new IMS release.
- Back up.
- UPGRADE.RECON—produces one good RECON.
- Back up new COPY1 RECON.
- Define new RECON.
- LIST.RECON STATUS or CHANGE.RECON DUAL to reinstate duality.
- Back up.
- Define SPARE RECON.

Note that UPGRADE.RECON only accepts a RECON1 DD statement and assumes that it is pointing at a valid copy. Use LIST.RECON STATUS to ensure that RECON1 is either COPY1 or COPY2. If it is not, use CHANGE.RECON REPLACE as described in Chapter 3, “RECON Care and Maintenance” to copy one of the current RECONs to RECON1.

No jobs that access RECON may be run during the UPGRADE process.

Version 6

Just to keep you on your toes, they changed it all in IMS V6 because the key length of the RECON records changes to accommodate the new UTC (year 2000) timestamps.

The procedure for V6 is documented in the *DBRC Guide and Reference.* Study it carefully and follow it religiously. There is no DOWNGRADE command—hence the recommendation to take BACKUPs at each stage of the process in case anything goes wrong.

Chapter 3 RECON Care and Maintenance

This chapter covers topics related to monitoring the status of your RECON data sets and what to do if something goes wrong with them.

Monitoring Status of RECONs

Loss or corruption of all RECON data sets is a major catastrophe, causing loss of service and probably jeopardizing system and data integrity. Adhering to the following guidelines will reduce the likelihood of RECON data sets being lost or corrupted.

1. Use the parameter STARTNEW(NO) on the INIT.RECON or CHANGE.RECON command.

2. Allocate a third spare RECON data set.

3. Use dynamic allocation.

4. Position all three data sets sensibly.

These precautions are important because only one RECON data set can be discarded without impacting availability.

Methods of Monitoring

Following are methods for monitoring RECON data sets in three different environments: online IMS system, batch, and CICS-DL/1.

Online IMS System

The status of the RECONs can be determined from any online IMS system using those RECONs (depending on terminal security for the /RML command) by entering the following:

```
/RML (LTERM xxxxxxxx) DBRC='RECON STATUS'
```

The optional LTERM parameter permits you to route the output from the RMLIST command to an alternative terminal (usually a printer). This method of monitoring the RECONs is only applicable while online IMS is running.

You can also issue online DBRC commands from automated operator programs. Many users set up batch message processing (BMP) programs that monitor the status of the RECONs regularly.

Batch

You can monitor the status of the RECONs in batch mode through a two-step job. First, issue a LIST.RECON STATUS command and write the file SYSPRINT to a data set. Second, analyze that output with a program. The advantage to this method is that it uses the standard dynamic allocation for RECONs.

CICS-DL/1

In CICS-DL/1, you can use the DBRC online transaction to issue the LIST.RECON STATUS command. However, DBRC commands are infrequently used in CICS-DL/1 because they require 430K additional virtual storage in the CICS-DL/1 region.

Satisfactory Status of the RECONs

RECON1, RECON2, and RECON3 should be of unequal size. If RECON3 is the largest, then RECON1 and RECON2 should be COPY1 and COPY2 (either order), and RECON3 should be the SPARE. A *SPARE* RECON is an empty VSAM KSDS.

```
IMS/ESA VERSION 6 RELEASE 1 DATA BASE RECOVERY CONTROL           PAGE 001

 LIST.RECON
 98.237  12:20:05.6        LISTING OF RECON                      PAGE 002

 RECON
  RECOVERY CONTROL DATA SET, IMS/ESA V6R1
  DMB#=20                        INIT TOKEN=97274F1724077F
  FORCER LOG DSN CHECK=CHECK44     STARTNEW=NO
  TAPE UNIT=CART     DASD UNIT=SYSALLDA   TRACEOFF  SSID=**NULL**
  LIST DLOG=YES             CA/IC/LOG DATA SETS CATALOGED=YES
  LOG RETENTION PERIOD=00.001 00:00:00.0

TIMESTAMP INFO......

 - DDNAME-      -STATUS-           -DATA SET NAME-
   RECON1        COPY1              IMS61A.RECON1
   RECON2        COPY2              IMS61B.RECON2
   RECON3        SPARE              IMS61C.RECON3
```

Figure 4 Output Showing Satisfactory RECONs

Reorganization of RECONs

Periodically, reorganization is necessary because the RECONs are VSAM KSDSs. DBRC automatically deletes some records; the user must delete others. See “GENMAX Cycle—Log Cleanup in RECON” on page 45 for additional information. After this cleanup, the user should perform a reorganization to reclaim free space and eliminate any CI and CA splits.

The Easy Way

The easiest way to reorganize a VSAM KSDS with the NOREUSE attribute is to REPRO the data set to a sequential (if record size is less than 32K) or VSAM data set, delete and redefine the KSDS, and REPRO the data back. However, this is not the safest method. With this method there will only be one valid RECON. If RECON(s) have been discarded, there may not be any valid RECONs.

A Safer Method

A safer method of reorganizing a VSAM KSDS is to successively use the CHANGE.RECON command with the REPLACE parameter, followed by the IDCAMS command to delete and define the discarded RECON.

You can write a program in a high-level language to determine the current status of the RECONs (see “Monitoring Status of RECONs” on page 21), invoke an Assembler routine that will link to DBRC (DSPURX00) and replace a RECON, and link to IDCAMS to delete and define the discarded RECON.

Note: It is not possible to delete a data set while it is allocated elsewhere. However, as far as I am aware (unless they have changed VSAM recently), you can delete a VSAM data set on one CPU while it is allocated on another CPU. This means that such a program cannot operate while any DBRC job is active.

An online IMS system will free a discarded RECON when it next accesses the RECONs. So you can write a BMP automated operator interface (AOI) program to check the STATUS of the three RECONs, issue the /RMCHANGE command to discard the first RECON, issue a /DIS OLDS or /RML RECON STATUS command to force IMS to access the RECONs and force deallocation of the now discarded RECON. Now run IDCAMS to delete and redefine the discarded RECON. You can repeat this procedure for the second RECON.

This procedure will work if this online IMS system is the only job currently using the RECONs. If other jobs are active, IDCAMS will have to wait until the other jobs have also discarded and deallocated the “bad” RECON. This can, for instance, be a problem when sharing the RECONs across CPUs.

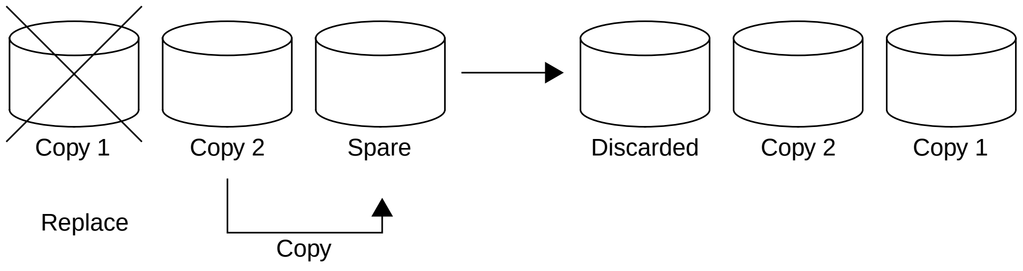


Figure 5 A Safer Method of Reorganizing the RECONs

If you issue a write-to-operator with reply (WTOR) or programming language one (PL/1) display after each CHANGE.RECON, you can also operate the program in a multi-online IMS environment. A message instructing the master terminal operator (MTO) to issue a /DIS OLDS to each online system before replying to the program would be displayed on job entry subsystem (JES) consoles.

/DIS OLDS

The /DIS OLDS command gives the status of all the OLDS and WADS—it accesses the RECONs as part of this process and hence can be used as outlined above to trigger the deallocation of a "bad" RECON.

```
OLDS-DDNAME  %FULL RATE ARCH-JOB  ARCH-STATUS  OTHER-STATUS

*DFSOLP05      37   9                          IN USE,BACKOUT
*DFSOLS05                                      IN USE,BACKOUT
 DFSOLP04                          NEEDED       BACKOUT
 DFSOLS04                          NEEDED       BACKOUT
 DFSOLP03                AR163456  SCHEDULED    WRITE-ERROR,STOP
 DFSOLS03                AR163456  SCHEDULED    STOPPED
 DFSOLP02                AR163456  SCHEDULED
 DFSOLS02                AR163456  SCHEDULED
 DFSOLP01                          AVAILABLE
 DFSOLS01                          AVAILABLE
 DFSOLP00                          AVAILABLE
 DFSOLS00                          AVAILABLE

 DUAL OLDS LOGGING, DUAL WADS LOGGING
 AUTOMATIC ARCHIVE = 02
 WADS = *DFSWADS0 *DFSWADS1 DFSWADS2 DFSWADS3
```

Figure 6 Sample /DIS OLDS Output

Current OLDS and WADS identified by *.

Recovery Following a Discarded RECON

An advantage to having a program that reorganizes the RECONs is that you can very easily design it to recover from a situation like a RECON being discarded. All you have to do is wait for the “bad” RECON to be discarded (= deallocated) by all the subsystems currently running and then use IDCAMS DELETE/DEFINE to create an empty VSAM KSDS. The next job step to access the RECONs will then find this data set and recognize it as a SPARE RECON. The trick is noticing that you have lost a RECON—either trap the RECON messages in an MVS exit or run a LIST.RECON STATUS on a regular basis to see if any of the RECONs is bad.

Losing Two RECONs

If you lose two RECONs and you have NONEW specified, then no new jobs will start. If you now issue LIST.RECON STATUS you will get output similar to the following:

```
DSP0014I    DYNAMIC ALLOCATION FAILED FOR RECON2
DSP0014I    RETURN CODE=04   REASON CODE=1708
DSP0014I    DYNAMIC ALLOCATION FAILED FOR RECON3
DSP0014I    RETURN CODE=04   REASON CODE=1708
DSP0243I    JOB TERMINATED DUE TO UNAVAILABLE SPARE RECON
DSP0243I    THE ONLY AVAILABLE RECON IS
DSP0243I    DSN=RCNBR.RVP.R61.RECON1
```

Figure 7 Output After Issuing LIST.RECON STATUS with Two Lost RECONs

Identify which is the correct valid RECON, delete/define the other two, and then issue CHANGE.RECON DUAL or LIST.RECON STATUS. This will reestablish dual RECONs with a SPARE.

Losing All Three RECONs

It is impossible to recreate all the information in the RECONs using commands. Restoring RECON to an earlier point in time and adding missing log etc. data is *not* a correct method (in the Recovery Control and Share Control environments). With careful planning, you should never lose all three RECONs.

The quickest and only safe method of rebuilding and re-synchronizing the RECON data sets in these circumstances is to clean up the abended jobs, which will involve backout of in-flight work, take a fresh set of RECONs with all the databases, groups etc. registered (it is worth keeping a PDS member handy with all these INIT definitions stored in it—if I ever get time, I will try and write a GENJCL.USER to create all the INIT commands from an existing RECON) and then image copy *everything*.

I met the man in charge of the IBM Red Books the other day, and he pointed out to me that there is a sample procedure for building the INIT.DB and INIT.DBDS commands from an existing RECON in SG24-2211—so that's saved me some work!

BACKUP.RECON

Why do I use BACKUP.RECON then, if not to cater for loss of the RECONs? BACKUP.RECON should still be used for several reasons:

1. Taking a copy of the RECONs offsite. Whenever you take data offsite (e.g. image copies and/or logs), you should take a copy of the RECONs at the same time—see Chapter 18, "Offsite Considerations" for further details.

2. Before applying maintenance—e.g. the Migration and Coexistence SPE—see "Changing Versions—the UPGRADE command" on page 19 for further details.

3. When something has gone wrong and you don't know what to do—take a BACKUP of the RECONs and play with them. When you've sorted out what commands to issue, go back and use the production RECONs.

BACKUP.RECON uses IDCAMS REPRO internally, but it also does an enqueue on the RECONs to guarantee that no updates occur during the BACKUP—a fuzzy backup is useless—so you should always use BACKUP.RECON rather than pack dumps or other software.

However, your RECON record size may be greater than 32K (e.g. SUBSYS record—see “Creating a RECON Data Set” on page 15 or PRILOG/PRISLD record—see “Size of PRISLD/PRILOG Record” on page 61) and REPRO will not copy a VSAM KSDS to a non-VSAM data set when the maximum record size is greater than 32,760 bytes. You have to back up the RECON to VSAM KSDS (use BACKUP.RECON so that you still get the enqueue, but point the output to an empty VSAM KSDS). When using this method of backup (rather than to non-VSAM data sets), you must use empty—IDCAMS DELETE/DEFINEd—KSDS. Otherwise, your output will be merged with the previous backup copy. Once you have backed up your RECONs to VSAM KSDS, then use another piece of software like DFDSS or FDR to copy this data set to tape/cartridge.

Chapter 4   Initializing RECONs

This chapter considers makes recommendations on some of the important parameters you have to set when initializing the RECONs.

INIT.RECON

To convert an empty VSAM KSDS into a RECON, you use the INIT.RECON command, which writes the RECON header records. These records identify the data set and are read whenever a RECON data set is opened. The parameters have the following meaning:

- **RECOVCTL**

  indicates that DBRC is to be used for database Recovery Control only—not recommended. Dies in V6.

- **SHARECTL**

  indicates that, in addition to Recovery Control, full DBRC Share Control is to be provided. SHARECTL is the default and is required for DBCTL.

- **NONEW**

  will only allow a new subsystem to start if two RECONs are available. This option is the default and is strongly recommended. If a RECON I/O error does occur, the spare can be brought into service to restore the situation where new jobs can be run.

- **STARTNEW**

  will allow a subsystem to start as long as there is one RECON available. This would lead to problems if that RECON also went down.

- **FORCER**

  specifies that you wish DBRC to force registration of all databases processed in jobs under DBRC's control, which means that if you run a job with one or more unregistered databases it will abend. This is your ultimate goal. This parameter also means that DBRC will check the data set name of your DBDS against the name registered in RECON and abend if they don’t match.

- **NOFORCER**

  means do not force registration of databases—default. In this case if the data set name does not match, you simply get a warning message—DB not registered. The typical case of this happening is when you have the same DBD names and ddnames for test and production but the data set names are different. You now run a test database against a production RECON by mistake—whoops!! Hence the recommendation to use FORCER.

Warning! The black hole in DBRC is when you run your production database against the test RECONs by mistake. You are very unlikely to be running FORCER on your test RECONs, so you would just get a message warning you that the database is not registered (and a message saying that you were running on the wrong set of RECONs—DFS487W) and the job would continue—you *must* make sure that you never run production databases against the wrong RECONs.

The DFS487W (and DFS485W, DFS486W) come from a fiendishly clever piece of DBRC code called DBRC Usage Indicator (DUI) checking, which is clever enough to tell if you are turning DBRC on and off, running on different RECONs etc. Unfortunately, it just gives you warning messages.

- **CHECKxx**

  tells DBRC which type of log tape data set name comparisons to use during its JCL verification process. CHECK17 is the default, and is there for historical reasons—remember MVS tape labels? (In all cases, the volume serial number is checked.) Use CHECK44.

- **SSID**

  Enter the name of the most frequently used online subsystem. This will save you having to enter the SSID on many commands during normal operation, since the default will be the one specified here.

- **CATDS|NOCATDS**

  are mutually exclusive, optional parameters you use to modify the status of whether image copy, change accumulation, and log data sets are catalog managed.

- **CATDS**

  specifies that these data sets are catalog-managed. If the data set is allocated by the catalog and the CATDS option is used, DBRC bypasses volume serial and file sequence verification for the data set. See "DBRC and Data Facility Hierarchical Storage Manager (DFHSM)" on page 62 for an explanation of why this is useful.

Note: In order for CATDS option to be effective, the data set must be cataloged, and volume serial information for the data set must be omitted from the JCL. If the data set is cataloged, CATDS is specified, and volume serial information is included in the JCL, DBRC ignores CATDS and allocates the data set by the JCL. Normal volume serial and file sequence checking occurs.

If the data set is not cataloged, CATDS is not effective, and DBRC allocates the data set by the JCL, with volume serial and file sequence checking.

- **NOCATDS**

  specifies that these data sets, whether cataloged or not, are not to be treated as catalog-managed. DBRC verifies that the volume serial and file sequence numbers appearing in the job file control block are the same as the information recorded in the RECON data set.

- **DASDUNIT/TAPEUNIT**

  specify default UNIT information for tape and DASD. DBRC identifies if a data set is on tape or DASD and stores that information in RECON. It does this for logs, image copy data sets, and change accumulation data sets. Every time you create one of these data sets, DBRC extracts a one-bit flag from MVS that indicates whether the data set is on tape or DASD. It then checks the RECON header record to see what you defined as the default tape and DASD units. So, if you have specified TAPEUNIT(3480), all data sets that are on tape will be stored in RECON with a UNIT of 3480. See "Concatenated Tape and Disk" on page 62 for details of how to use these parameters.

- **LOGRET**(time_interval)

  is an optional parameter you use to specify the retention period for log data sets. The retention period is the minimum amount of time in which a log becomes inactive after it is opened. (It is then eligible to be deleted).

- **LISTDL|NOLISTDL**

  are mutually exclusive, optional parameters you use to specify whether data set names deleted from the RECON (by DELETE.LOG command or by an archive job log compression) are listed in the job output. The setting specified on this command can be overridden by the DELETE.LOG command. There is no way to override the setting for log compression (see "Continuous Operations" on page 58 for more information) during an archive job.

- **UPGRADE**

  is an optional parameter you use to specify that the RECON is to be upgraded for IMS V6. It is required whenever a new V6 level RECON data set must be created for use by the Upgrade utility. This parameter sets a safety switch that makes the new RECON unusable until the Upgrade utility is run. Likewise, the upgrade utility requires that the switch has been set in the new RECON.

Entering Timestamps

Prior to V6, timestamps in RECON are 12 digits in the form

```
yyydddhhmmsst
```

V1, 2, 3, and 4 will not run in the year 2000. V5 (with the correct PTFs—see BMC Software manual *Year 2000 and IMS*) will.

This is how you would enter a timestamp in a command:

```
GENJCL.RECOV DBD(...) DDN(...) RCVTIME(981231014169)
```

and when you print out the RECONs (or preferably just bits of them as LIST.RECON tends to kill trees), DBRC will format these for you:

```
98.123 10:14:16.9
```

Some people have purchased or written ISPF front-ends to DBRC to enable them to enter commands and the timestamps more easily. I personally think this is a waste of time—my objective is to enter a “native” DBRC command as infrequently as possible. To me, the interface to DBRC should be automated, and if I need to find a timestamp/build a DBRC command on a regular basis, then I buy a product and/or write a GENJCL.USER procedure to extract the information from RECON and build the commands/JCL for me automatically.

Version 6

V6 uses Coordinated Universal Time (UTC—I know the initials are in the wrong order, but this comes from the French). UTC is also commonly referred to as Greenwich Mean Time (GMT—correct order, it’s British!). Actually, they are wrong—GMT and UTC are not the same—the difference is leap seconds. Is this important? I have no idea.

In V6 you can enter timestamps in one of the following formats:

- compressed: yydddhhmmsst [offset]
- punctuated: [yy]yy/ddd/hh/mm/ss/t [offset]

  where:

yyyy	is the year (0000 to 9999)
ddd	is the day (000 to 365)
hh	is the hour (0 to 23)
mm	is the minute (0 to 59)
ss	is the second (0 to 59)
t	is the tenth of a second (0 to 9)
/	can be any non-numeric character delimiter including blank except the apostrophe.

  Offset can be one of the following:

  — Omitted. The current TIMEZIN value (see below) will be used.

— A numeric offset in the form *hh:mm* or *hhmm* that, when added to UTC, gives local time. *hh* is a numeric value from 0 to 14. For the compressed format if *mm* is specified, then *hh* must also be specified. *mm* is a value from the set {00, 15, 30, 45}.

hh:mm is valid only between the values -11:45 to +14:45.

hhmm is valid only between the values -1145 to +1445.

— A symbolic time zone label.

The value may have elements truncated on the right, in which case the omitted element's digits are assumed to be zeros.

You may truncate the input at the beginning of any element after *ddd*; so, *yyyy/ddd* is acceptable as is *yyyy/ddd/hh*. Part of an element may not be entered: for example, *yyyy/ddd/h* is invalid.

The same time stamp value could be entered in the following ways:

```
942520824457                942520824457-0800

94.252 08:24:45.7 -8          94/252-08.24.45.7 -8:00

94,252 08:24:45.7 PST          94.252/08:24:45.7

94/252-16.24.45.7 UTC          1994 252 16.24.45.7 +0
```

Note: See also the BMC Software manual *Year 2000 and IMS* for a description of the timestamp changes between IMS V5 and V6 and how these are stored in IMS log records, RECONs, etc.

The Time History Table (V6 only)

A time history table (THT) is created each time INIT.RECON is issued in IMS V6. The THT in the V6 RECON contains a history of time-zone changes. This is basically a record of when the system time was changed to and from Daylight Savings Time. The THT is used

- in the RECON Upgrade process (PGM=DSPURU00)
- in the RECON Coexistence environment to enable pre V6 DBRC code, with the proper RECON Migration and Coexistence support applied, to convert pre-V6 timestamps to the UTC format

You need the THT to upgrade a V4 or V5 RECON to V6. If you share a V6 RECON with a V5 and V6 system, then you need the THT—this is “coexistence” and you most likely upgraded the V5 RECON to a V6 RECON by following the RECON upgrade procedure. To coexist, you need the RECON Migration and Co-existence APAR (PN89491 or PN89490) applied. With it, DBRC “downgrades” timestamps it reads from the V6 RECON to the form for processing and then “upgrades” the 7-byte form to UTC format using the THT. This timestamp processing affects all DBRC processing (including Database Recovery).

Note: The execution of the V6 INIT.RECON command will create a THT record. You create the THT with the INIT.RECON command in V6. You change it with the CHANGE.RECON THT (or REPTHT) command in V6.

You change the THT whenever you change the system clock and you tell DBRC by issuing the CHANGE.RECON THT command. If you are not coexisting with pre V6 DBRCs and never intend to—i.e., you do not need a “fallback” capability—then you do not need to worry about the THT.

- **THT|REPTHT**

  are mutually exclusive, optional parameters you use to specify whether an additional new time history table is created or the existing entry is replaced when the current THT entry and current MVS offset differ.

- **THT**

  generates a new THT entry if the current MVS offset from UTC and the current THT entry differ.

- **REPTHT**

  deletes the current THT entry and generates a replacement if the current MVS offset and the current THT entry differ. REPTHT is not valid if the table contains only the original single entry.

In other words, if your RECONs are *only* being used by V6, then you can forget all about the THT. If your RECONs are being shared between V6 systems and V4/5 systems, then you need the THT.

Warning! The code that allows you to keep IMS running when you change the clocks *only* applies when you are running V6—you cannot change the clocks and keep IMS running if you still have V4/5 systems using the RECONs. If you do, you will probably end up with bad timestamps and totally confused RECONs—see Appendix A, “THT Problems APAR—PQ03038” for details on how you can make a complete mess. The moral is don’t change the clocks and keep IMS running until you are *totally* on V6.

- **TIMEZONE**((label offset),(label offset))

  alters the time zone label table. This parameter is used to define one or more symbolic time zone labels because most people do not readily associate a numeric offset with a time zone. TIMEZONE allows you to define symbolic labels, like *PST* (Pacific Standard Time), for numeric offsets, such as *-8*. The time zone label table can contain up to 32 entries each composed of a label and an offset.

  Adding, replacing, and deleting entries from the stored list is supported as follows:

  — Adding an entry to the stored table is accomplished when an input list entry contains both a label which does not exist in the RECON and a valid offset value.

  — Replacing an entry to the stored table is accomplished when the input entry contains both a label that matches an existing label in the table and a valid offset value.

  — Deleting an entry to the stored table is accomplished when the input entry is a label that matches an existing label in the table and no offset value. If the offset is omitted, and the label is not found in the table, the table is not altered.

  The labels in the table must be unique.

- **TIMEZIN**(offset_rep [,duration])

  an optional parameter you use to define a default time zone value for time stamps entered without time zone information on subsequent DBRC commands.

- offset_rep

  The default time zone value. It can be one of the following choices:

  - *label* means a time zone label that has been previously defined using the TIMEZONE parameter.

  - *offset* means a numeric offset value in the same form as defined above for the TIMEZONE parameter.

  - *%SYS* is a keyword used to designate that the offset is to be derived from the current offset found in the MVS CVT control block. This is the initial default for DBRC.

- duration

  Specifies the duration of the offset_rep choice:

  - *PERM* indicates that the label or offset default is to be in effect for any subsequent DBRC command running with the same RECON.

  - *TEMP* indicates that the label or offset default is in effect only for the job in which the command is entered. If neither PERM nor TEMP is coded, TEMP is assumed.

- **TIMEFMT**(sublist)

  an optional parameter you use to define the form in which timestamps appear in messages, displays, and listings from DBRC.

  The format of the TIMEFMT parameter sublist is as follows:

  TIMEFMT(offset,offset_display,form,year_size,duration)

  If any of the items in the sublist is omitted, the current value from the RECON header is used.

  - offset

    specifies the offset which is to be applied to the UTC internal time before display:

    - *U* means none—that is, display UTC when the event occurred.

    - *O* means offset of origin—that is, display local time when and where the event occurred.

- *L* means current local offset—that is, display the current-local-time equivalent.

— offset_display

specifies the display format of the offset which is appended to the time:

- *L* specifies that the offset is to be displayed in label format, if a label has been defined for it. If no label is defined, the offset is displayed in numeric format.
- *O* specifies that the offset is to be displayed in the numeric (+/-HH:MM) format.
- *N* specifies that no time zone information is to be displayed.

— form

specifies whether the timestamp is to be displayed in punctuated or compressed form:

- *P* specifies that the timestamp is to be displayed in punctuated form.
- *C* specifies that the time is to be displayed in compressed form.

— year_size

specifies, for the punctuated form, whether all four digits of the year are to be displayed or only the two low-order digits. (This specification is ignored when the form is compressed; the compressed form only displays two digits.)

- *2* means only the units and tens digits of the year are displayed.
- *4* means all four digits of the year are to be displayed.

— duration

specifies the scope of these choices to be either limited to the current job or global overrides to the system defaults:

- *PERM* indicates that the specified options are to be in effect for any subsequent DBRC utility job running with the same RECON, i.e. these values become the defaults for subsequent jobs.

- *TEMP* indicates that the specified options are in effect only for the job in which the command is entered. If neither PERM nor TEMP is coded, TEMP is assumed.

Besides its use on the CHANGE.RECON command, the TIMEFMT parameter can be coded on any LIST.xxx or GENJCL.xxx DBRC command. It can also be specified in a skeletal JCL member as:

```
%SET TIMEFMT(....)
```

Here is an example of the %SET keyword in skeletal JCL.

```
%SET TIMEFMT(,N)
%SELECT RLDS(%SSID,LAST)
LOGEND =%LOGETIM
%ENDSEL
```

Figure 8 Example of %SET TIMEFMT

And here is what the output from the preceding example of %SET would render:

```
LOGEND =960111315000
```

The values set in the RECON by INIT.RECON or the upgrade utility are TIMEFMT(O,N,P,2).

The defaults used by GENJCL commands are TIMEFMT(O,O,C,2).

Example of it Going Wrong

To show you what this means in practice, imagine you issued the following command:

```
GENJCL.CA GRPNAME(AZB38PHD) NOJOB NOLIST MEMBER(CAJCL) -
CATIME('20000011355000')
```

This won’t work, because DBRC thinks you should be using two-digit years on GENJCL (default TIMEFMT above). You will get the following messages:

```
DSP0304I  TIMER SERVICES FAILURE
DSP0304I  FUNCTION = EXTERNAL TO INTERNAL
DSP0304I  DIAGNOSTIC CODE=DDD
DSP0304I  CALLER=DSPURX00 TIMESTAMP=
DSP0204I  INVALID VALUE FOR PARAMETER CATIME
DSP0209I PROCESSING TERMINATED WITH CONDITION CODE = 12
```

After playing around, we finally figured out that in order to specify a CATIME in punctuated format with a full four-digit year and have the command run successfully, the timestamp must be specified like this:

CATIME('2000 001 13 55 00 0') or CATIME(2000/001/13/55/00/0)

Alternatively, as GENJCL is expecting two-digit years you could use:

```
GENJCL.CA GRPNAME(AZB38PHD) NOJOB NOLIST MEMBER(CAJCL) -
CATIME(000011355000)
```

```
DSP0203I  COMMAND COMPLETED WITH CONDITION CODE 00
DSP0220I  COMMAND COMPLETION TIME 2000.002 01:56:43.4
IMS/ESA VERSION 6 RELEASE 1  DATA BASE RECOVERY CONTROL
DSP0211I  COMMAND PROCESSING COMPLETE
DSP0211I  HIGHEST CONDITION CODE = 00
```

Or you could specify a different TIMEFMT on the GENJCL command. I would personally carry on using 12-digit timestamps like we have all been using for years.

```
RECON
 RECOVERY CONTROL DATA SET, IMS/ESA V6R1
 DMB#=9                            INIT TOKEN=97274F1724077F
 NOFORCER  LOG DSN CHECK=CHECK17    STARTNEW=NO
 TAPE UNIT=3400      DASD UNIT=SYSDA     TRACEOFF   SSID=**NULL**
 LIST DLOG=YES                CA/IC/LOG DATA SETS CATALOGED=NO
 LOG RETENTION PERIOD=00.000 00:15:00.0

  TIME STAMP INFORMATION:
  TIMEZIN = %SYS             -LABEL- -OFFSET-
                              PDT     -07:00
                              PST     -08:00

  OUTPUT FORMAT:  DEFAULT = LOCORG NONE   PUNC YY
                  CURRENT = LOCAL  OFFSET PUNC YYYY

-DDNAME-      -STATUS-       -DATA SET NAME-
 RECON1        COPY1          IMSTESTL.IMS.RECON1
 RECON2        COPY2          IMSTESTL.IMS.RECON2
 RECON3        SPARE          IMSTESTL.IMS.RECON3


THT
        -LOCAL START-   -OFFSET-
     0000.000 00:00:00.0 +00:00
```

Figure 9 Sample RECON Header Record and Time History Table

Chapter 5 DBRC Registration

Registration is for physical DL/I and Fast Path data entry databases, including VSO DEDBs. There is no support in DBRC for main storage databases (MSDB) or generalized sequential access method (GSAM).

Partitioned databases are registered to DBRC as databases with multiple data set groups (DSgroups). In other words, there will be one INIT.DBDS for each partition. You cannot combine DSgroups and partitions in one database. This means that DBRC does not actually recognize the existence of partitions. As far as DBRC is concerned, it is one database; while you can copy or read partitions in parallel, you cannot run updating utilities against partitions in parallel as DBRC only allows utilities to share at the database level, not at the block level. If you want to update multiple partitions in parallel with batch jobs, then you must run these as BMPs or use n-way sharing.

The BMC Software utilities can copy, recover, and reorganize partitions in parallel—see Chapter 12, “BMC Software and DBRC” for more details of these utilities.

Database Registration

In order for DBRC to use Share Control (or Recovery Control if you are a coward) for a particular database, you must tell DBRC which databases you want it to control. This is done with two commands: INIT.DB and INIT.DBDS (plus INIT.ADS for Fast Path).

INIT.ADS provides the extra area information required for Fast Path. If you are using multiple area data sets, you issue one INIT.ADS for each copy of the area, minimum one, to a maximum of seven. See later in this chapter for more details on Fast Path DEDB registration and a sample MADS record.

The INIT.DB command is used to define a database and provide data sharing information. The INIT.DBDS command is used to define the database data sets and to provide recovery related information. The SHARELVL parameter on the INIT.DB command specifies the maximum level of sharing desired:

- Level 0 means that DBRC will ensure no sharing occurs.
- Level 1 means database level sharing is allowed.
- Level 2 means block level sharing in one MVS image.
- Level 3 means block level sharing across multiple MVS images.

```
DB
   DBD=DBRVI33     IRLMID=*NULL       DMB#=53     TYPE=IMS
   SHARE LEVEL=0  GSGNAME=**NULL**    USID=0000000002
   (USID info)
   FLAGS:                          COUNTERS:
      BACKOUT NEEDED        =OFF    RECOVERY NEEDED COUNT   =0
      READ ONLY             =OFF    IMAGE COPY NEEDED COUNT =0
      PROHIBIT AUTHORIZATION=OFF    AUTHORIZED SUBSYSTEMS   =0
      RECOVERABLE           =YES    HELD AUTHORIZATION STATE=0
                                    EEQE COUNT              =0
     (TRACKING info)
```

Figure 10   Sample DB Record

```
DBDS
 DSN=RVP.QA.DBRVI33                            TYPE=IMS
 DBD=DBRVI33   DDN=DBRVI33  DSID=001   DBORG=INDEX  DSORG=VSAM
 CAGRP=**NULL**  GENMAX=3   IC AVAIL=0 IC USED=0  DSSN=00000001
 NOREUSE         RECOVPD=0
 DEFLTJCL=**NULL**  ICJCL=ICJCL  OICJCL=OICJCL  RECOVJCL=RVPJCL
 FLAGS:                          COUNTERS:
  IC NEEDED     =OFF
  RECOV NEEDED  =OFF
  RECEIVE NEEDED=OFF                EEQE COUNT             =0
```

Figure 11   Sample DBDS Record

Some of the parameters in the INIT.DBDS command are as follows:

- **RECOVABL/NONRECOV**

  The default for databases is that they are recoverable—in other words that you will use image copies plus logs/change accumulations to recover them.

However, there are some databases that you may never want to recover this way, e.g. work databases that you simply reinitialize. NONRECOV is designed for these. (You cannot specify NONRECOV for a Fast Path DEDB.) It is also designed for anyone who has been clever enough to buy BMC Software's SECONDARY INDEX UTILITY/EP product, which can rebuild primary and secondary indexes without needing an image copy and logs.

If you register a database as NONRECOV, you are not required to image copy it and DBRC will not record any ALLOC/LOGALL information for it in RECON. As far as DBRC is concerned, the database never gets updated. To recover it, you will reload it, reinitialize it, or use SECONDARY INDEX UTILITY/EP, or take one image copy at the beginning and use GENJCL.RECOV RESTORE to restore it from this original image copy.

IMS will write *before* images to the log in case backout is required, but it writes no *after* images. At archive time, the *before* images are not copied across to the SLDS or RLDS for NONRECOV databases. This means that dynamic backout and /ERE backout and batch backout from batch logs perform normally, but you cannot run batch backout for NONRECOV databases in an online system (e.g. after /ERE failure) from a SLDS—this can only be done from the OLDS. This can also have implications on disaster recovery procedures—see Chapter 18, "Offsite Considerations" for further details.

- **GENMAX**

  Use this to say how many generations of image copy data sets you want DBRC to remember for this particular database data set. For example, if you set GENMAX to three, DBRC keeps information about the latest three image copies for this database data set in RECON.

- **REUSE/NOREUSE**

  This parameter dictates how you are going to allocate your image copy output data sets.

  REUSE tells DBRC you are going to predefine GENMAX image copy data sets for later use by the image copy utility. DBRC reuses these data sets in a round-robin fashion. This is designed for pre-allocated data sets on DASD. If you use REUSE, issue INIT.IC commands for the number of generations you wish DBRC to record, i.e., if GENMAX=3, then register three image copy output data sets in RECON for this database data set. If they are on DASD, you must physically allocate them using IEFBR14. You should use GENJCL.IC to generate your image copy JCL with the correct output data sets. Your skeletal JCL will contain:

```
//DATAOUT1 DD DSN=%ICDSN1,.....
//DATAOUT2 DD DSN=%ICDSN2,.....
```

Figure 12 Image Copy Skeletal JCL with REUSE

DBRC fills in the appropriate data set names, VOLSERs, etc., from the information you have stored in RECON via the INIT.IC commands.

NOREUSE tells DBRC that there are no predefined data sets in RECON. DBRC picks up the information from your JCL when you run the image copy and stores the information in RECON. This is the way to use DBRC with GDGs. Define NOREUSE and do *not* issue any INIT.IC commands. Use the same JCL every night (you won't use GENJCL.IC). Your JCL will look like this:

```
//DATAOUT1 DD DSN=YOURGDG.DSNAME.COPY1(+1),.....
//DATAOUT2 DD DSN=YOURGDG.DSNAME.COPY2(+1),.....
```

Figure 13 Image Copy Skeletal JCL with NOREUSE

DBRC stores the fully expanded GDG name in RECON after successful completion of the utility.

- **RECOVPD**

  With the RECOVPD parameter, you are instructing the system to retain image copy data sets until they are over *n* days old. The parameter interacts with GENMAX and REUSE/NOREUSE. It is best explained with an example.

  Say you have GENMAX set at 3 and RECOVPD at 20 days. If you run the fourth image copy when the first image copy is only 15 days old, the first image copy is not thrown away. If you specified NOREUSE, DBRC dynamically changes GENMAX from 3 to 4 and does not automatically change it back later. One customer has implemented a procedure that tries to reset GENMAX at the end of each day; DBRC will only reduce GENMAX and delete the extra generations if they are outside the RECOVPD. Alternatively, you can have a look at my *GENJCL.USER Examples and Description* manual (top of the Falkland Islands reading list last winter) for an example of cleaning up GENMAX.

If you specify REUSE, DBRC does not find a fourth generation in RECON and the job fails. You then have to change GENMAX, issue another INIT.IC command, allocate the output data set(s), and rerun GENJCL.

GENMAX Cycle—Log Cleanup in RECON

When GENMAX is exceeded (e.g., you have set it at three image copies and you create a fourth image copy that causes the first one to be discarded), many cleanup operations occur. The oldest image copy is deleted. Now IC2 is the oldest image copy for this database. DBRC deletes any ALLOC records that occurred prior to what is now the oldest image copy (IC2). The same cleanup occurs for reorganization records and recovery records.

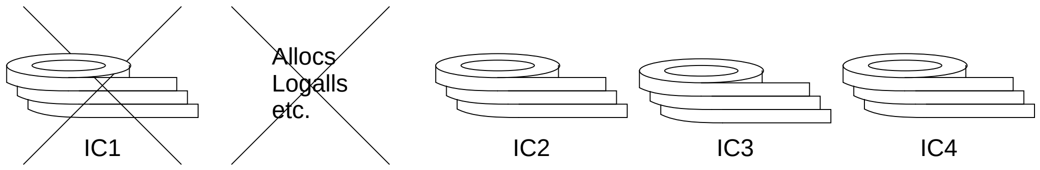


Figure 14 Log Cleanup in RECON

When ALLOC records are discarded, DBRC performs maintenance on the LOGALL record associated with the PRILOG record referenced by the ALLOC records. Records that are associated with logs (i.e., PRILOGs, PRISLDs, etc.) are not automatically deleted. To delete these records, DBRC control statements must be issued, followed by a RECON reorganization. See “Reorganization of RECONs” on page 23 for additional information.

The correct command to use in a Recovery Control or Share Control environment is DELETE.LOG INACTIVE—see “DELETE.LOG INACTIVE” on page 57 for more details.

Size of RECONs

If you are using DBRC Recovery Control or Share Control, RECON must keep records for the maximum value of

GENMAX x Image Copy Frequency

If you have a database you copy once per month and you have an installation default of six generations, the RECONs must be large enough to hold six months worth of recovery-related information. Keep your RECONs small—they are a potential bottleneck, especially when shared between multiple jobs/MVS images. Do not submit 500 image copy jobs at the same time—the enqueues will be horrendous. Group databases together in job steps, and design your jobs to “get in and out” of the RECONs as fast as possible.

Change Accumulation Registration

```
CAGRP
 GRPNAME=MYGROUP  GRPMAX=3   CA AVAIL=2 CA USED=1
 REUSE   CAJCL=CAJCL         DEFLTJCL=**NULL**
                             #MEMBERS=5    -DBD-      -DDN-
                                           HDAMOSAM   HDAMOSAM
                                           HDAMVSAM   DD1
                                           HDAMVSAM   HDAMVSAM
                                           HISAMVSA   HISAMVSA
                                           HISAMISA   HISAMISA
```

Figure 15 Sample CAGRP Record

```
CA
 DSN=IMSVS.MYGROUP.CA.CA1                        FILE SEQ=1
 CAGRP=MYGROUP      STOP    = 1998.274   08:40:43.5 - 09:00*
                    UNIT=CART     VOLS DEF=2   VOLS USED=2
                                 VOLSER=ABC123
 RUN     = 1998.274   09:26:19.2 -09:00
  DBD=HDAMOSAM DDN=HDAMOSAM PURGETIME=1998.274 08:38:21.4 -09:00
    CHANGES ACCUMULATED=YES COMPLETE CA=YES INDOUBT EEQES=NO
    LSN    = 000000000000      DSSN   = 0000000001
    LRID   = 0000000000000414  USID   = 0000000002
   etc. one entry for each member of CAGRP
```

Figure 16 Sample CA Record

Use the INIT.CAGRP to register CAGRPs in RECON. Some of the parameters are as follows:

- **GRPMAX**

  GRPMAX is similar to the GENMAX parameter on the INIT.DBDS command. GRPMAX specifies how many generations of change accumulation data sets you want to maintain for this group.

- **REUSE/NOREUSE**

  These are similar to the parameters specified for image copies. REUSE allows you to predefine to DBRC a number of data sets (using INIT.CA commands, followed by IEFBR14) to be used for output for this change accumulation group in a round-robin fashion. You should use GENJCL.CA. Your skeletal JCL will contain the following:

```
//DFSUCUMO DD DSN=%caodsn,.....
//DFSUCUMN DD DSN=%candsn,.....
```

Figure 17 Change Accumulation Skeletal JCL with REUSE

  DBRC fills in the appropriate data set names, VOLSERs, etc., from the information you have stored in RECON using the INIT.CA commands.

  NOREUSE tells DBRC there are no predefined data sets in RECON. DBRC picks up the information from your JCL when you run the change accumulation and stores it in RECON. This is the way to use DBRC with GDGs. Define NOREUSE and do not issue any INIT.CA commands.

  Your JCL will not be the same every time because the logs selected will be different. So use GENJCL.CA and set up your skeletal JCL to look like this:

```
//DFSUCUMO DD DSN=%caodsn,.....
(let DBRC select from RECON)
//DFSUCUMN DD DSN=YOURGDG.DSNAME.CA(+1),.....
```

Figure 18 Change Accumulation Skeletal JCL with NOREUSE

DBRC stores the fully expanded GDG name in RECON after successful completion of the utility.

It is possible to generate multi-step jobs with DBRC. For example, you can use the VOLNUM parameter on GENJCL.CA to create a new step each time the VOLNUM number of input log volumes is exceeded. If you use GDGs, this can cause problems as DBRC generates the same GDG numbers in each step. The solution to this is to use the %SET keyword in your skeletal JCL to select a new skeletal JCL member for the next step. For example, skeletal JCL member 1 finishes with a %SET for member 2. Member 1 uses GDG(0), member 2 uses GDG(+1), etc. See Appendix C, “DBRC GENJCL Enhancements” for additional information.

DSLOG

Do NOT use this facility—it is slow, inefficient, a nightmare for the data administrator, and is no longer supported in IMS V3, and oh yes I absolutely hate(d) it!

Reordering DSGroups

You cannot reorder data set groups in a database without deleting the database from the RECONs. Unload the database, DELETE.DB it from the RECONs, re-register it, and then reload it.

At registration time, DBRC assigns a data set ID to each DBDS record. If you change the order of the data set groups, reorganization gets confused—“Data set identifier in DBDLIB and RECON do not match”.

DBDS Groups and DB Groups

A DBDS group is a named collection of DBDSs or DEDB areas. DBRC can perform various operations by DBDS group so that you do not have to repeat the command for every member of the group.

```
INIT.DBDSGRP GRPNAME(name) MEMBERS((dbname,ddname),..)
```

When you specify a DBDS group on a command, DBRC invokes that command for each member of the DBDS group. Groups can be used on the LIST command and GENJCL.USER command—have a look at my *GENJCL.USER Examples and Description* manual for some examples.

You can also specify a CA group as a DBDS group. DBRC then executes the command for each member of the CA group.

You can define as many DBDS groups as you wish. Up to 1024 DBDSs can be in a group (2000 in IMS V6). All DBDSs in a group must be registered in RECON. A DBDS can belong to more than one DBDS group.

A DBDS group can be “implied.” An implied DBDS group exists for each database registered in RECON. The members of this implied group consist of all DBDSs of the database. To use an implied group, specify the database name without specifying a ddname on the above GENJCL and LIST commands.

When using DBDS groups, DBRC holds and releases RECON for each execution of the command for each member DBDS. As a result, integrity of the output is preserved for each DBDS but may not be preserved across the group. This is because RECON may have changed between its last release and hold. The trade-off here is that performance could be adversely affected if DBRC were to hold RECON while processing an entire DBDS group.

A DB group is a named collection of databases or DEDB areas. A DB group name can be specified in the /START, /STOP, and /DBRECOVERY commands instead of issuing these commands separately for each database or area. This greatly reduces the number of times these commands must be issued. Use the DATAGROUP keyword to specify the DB group name.

INIT.DBDSGRP GRPNAME(name) DBGRP(dbname,..)

You can define as many DB groups as you wish. Up to 1024 databases or areas may be in a group (2000 in IMS V6). A database or area can belong to more than one DB group and need not be registered in RECON.

Note: Although a DBDS group can be used as a DB group, you should use a database group whenever possible. Processing a DBDS group as a DB group entails increased overhead.

BMC Software’s RECOVERY MANAGER for IMS product offers a variety of interactive grouping capabilities, including validation and ease of maintenance.

Fast Path Considerations

Fast Path users are recommended to read the *Guide to IMS/VS Version 1.3 DEDB Facility* manual, GG24-1633. This is an excellent and comprehensive manual. This chapter only contains a few summary points.

Registration

There are some extra parameters on the INIT.DBDS command for Fast Path DEDBs:

- **PREOPEN|NOPREO**

  are mutually exclusive, optional parameters you use to specify whether an area is to be opened after the first checkpoint following the next control region initialization or when the next /START AREA command is processed. The default is NOPREO. If you specify PRELOAD, then PREOPEN is the default.

- **VSO|NOVSO**

  are mutually exclusive, optional parameters you use to specify whether an area will reside in virtual storage the next time the control region is initialized or when the next /STA AREA command is processed.

- **CFSTR1(name)**

  is an optional parameter you use to specify the name of the first coupling facility structure for the identified area.

- **CFSTR2(name)**

  is an optional parameter you use to specify the name of the second coupling facility structure for the identified area.

- **LKASID|NOLKASID**

  are mutually exclusive optional parameters you use to specify whether local data caching for the specified area is used for buffer lookaside on read requests.

- **PRELOAD|NOPREL**

  are mutually exclusive, optional parameters you use to specify whether a VSO area is to be loaded the next time it is opened.

The other differences from the DL/I DB specifications already discussed are as follows:

- **INIT.DB**

  for DEDBs, code the additional parameter TYPEFP. The default is TYPEIMS.

- **INIT.DBDS**

  Instead of coding DDN(name), you should code AREA(name).

- **INIT.ADS**

  Fast Path DEDBs can have multiple area data sets (i.e., multiple copies of the same areas). To provide DBRC with the necessary information to control the multiple data sets, this command must be issued once for each copy of each area registered (even if you are only using single area data sets).

If a Fast Path database is registered in RECON, then IMS can use this information for dynamic allocation—there is no need to run the DFSMDA macro for them.

Multiple Area Data Sets

MADS allow you to continue accessing the whole of an area, even if one or more CIs are bad (read from another copy). It requires DBRC Recovery or Share Control. The area data sets must be registered in RECON using the INIT.ADS command.

New utilities:

- MADS Compare and MADS Create provide in-flight recovery.
- Concurrent Image Copy provides in-flight copy.

  Image copy runs as long as there is at least one good copy of each CI available. Bad CIs are indicated via the presence of EQEs in the second CI of the area data sets. A maximum of ten are allowed before the area data set stops.

```
----------------------------------------------------------------------
DB
 DBD=DEDBJN21                                     DMB#=9        TYPE=FP
 SHARE LEVEL=1
 FLAGS:                            COUNTERS:
                                     RECOVERY NEEDED COUNT   =2
                                     IMAGE COPY NEEDED COUNT =0
   PROHIBIT AUTHORIZATION=OFF        AUTHORIZED AREAS        =0
                                     EEQE COUNT              =0
----------------------------------------------------------------------
DBDS
 DBD=DEDBJN21  AREA=DB21AR0                                     TYPE=FP
 SHARE LEVEL=1             DSID=001 DBORG=DEDB   DSORG=VSAM
 GSGNAME=IMSGSG1              USID=0000000002
 AUTHORIZED USID=0000000002  RECEIVE USID=0000000002 HARD USID=0000000002
 RECEIVE NEEDED USID=0000000000
 CAGRP=**NULL**  GENMAX=2     IC AVAIL=0     IC USED=1     DSSN=00000001
 NOREUSE         RECOVPD=0      NOVSO  NOPREOPEN  NOPRELOAD
 CFSTR1=**NULL**         CFSTR2=**NULL**          NOLKASID
 DEFLTJCL=**NULL**  ICJCL=ICJCL    RECVJCL=ICRCVJCL  RECOVJCL=RECOVJCL
 FLAGS:                            COUNTERS:
   PROHIBIT AUTHORIZATION=OFF        AUTHORIZED SUBSYSTEMS   =0
   HELD AUTHORIZATION STATE=0
   IC NEEDED            =OFF         ADS AVAIL #             =2
   RECOV NEEDED         =OFF         REGISTERED ADS #        =2
   DATABASE LEVEL TRACK  =YES        EEQE COUNT              =2
   RECEIVE NEEDED       =OFF
   OFR REQUIRED         =NO
   TRACKING SUSPENDED   =NO
   HSSP CIC IN PROGRESS  =NO
ERROR QUEUE ELEMENTS:
   -EQERBA-  -EEQE TYPE-  - SSID -
   00004000   INDOUBT       SYS1
   00008000   INDOUBT       SYS1

 ADS LIST:
                                                         CREATE
    -ADS DDN--ADS DSN-                           -STAT- -RUNNING-
   DB23AR11 ADS1.DB23AR1                          AVAIL    NO
   DB23AR12 ADS2.DB23AR1                          AVAIL    NO
--------------------------------------------------------------------
```

Figure 19 Sample MADS Record

Flags and Counters

The following charts show how the Fast Path utilities and DBRC commands interact with the flags in RECON:

Command	Record	Flag/Status
INIT.DB	DB	
INIT.DBDS	DBDS	Recovery Needed is set on
INIT.ADS	DBDS	Unavailable status is set
DEDB	DBDS	Recovery Needed must be set on (start)
Initialization	DBDS	Recovery Needed is set off (end)
Utility	DBDS	Available status is set

Figure 20 Interaction of Flags and Commands for Fast Path

	DEDB Init		Image Copy		Full Recovery	
	Start	End	Start	End	Start	End
PFA flag in DB record	Bypassed		Bypassed		Bypassed	
PFA flag in DBDS record	Bypassed		Bypassed		Bypassed	
Recovery Needed in DBDS record	On	Off	Off		On	Off
ADS status	Unavail.	Avail.			Unavail.	Avail.
Image Copy Needed in DBDS record	Bypassed	On	Bypassed	Off	Bypassed	

Figure 21 Interaction of Flags and Utilities for Fast Path

- PFA = Prohibit Further Authorization

High Speed Sequential Processing (HSSP)

This was a new feature introduced in IMS V3; it allows batch-type programs (BMPs) to do track I/O against a DEDB instead of block I/O (gosh, there’s a good idea, I wonder where I’ve seen that before?!). It could also take an image copy as it was working its way through the area. Internally it was using the cache in the DASD control units, and it transpired that the hardware gave you the performance improvements and the HSSP benefit was unpredictable and often not significant.

Now, no one understood how all this worked and I never met anybody who used it. Fortunately, it all got totally rewritten in IMS V5, with the result that it is now simple, comprehensible, and works. The principle is still the same, as you work your way through the database with a BMP, it takes an image copy at the same time and registers this in RECON. However, nowadays it is using software techniques and central storage to achieve performance improvements, with chained reads, look-ahead reading etc.

The V5 HSSP Image Copy option is also totally new. It now runs asynchronously—in parallel with and slightly behind HSSP application processing—and creates a standard image copy.

HSSP requires the DEDB to be registered as REUSE if you want to use the Image Copy feature. Image copies created by HSSP are Concurrent Image Copies.

One other thing to watch out for with HSSP image copies is that the term “sequential” must be taken literally. If you try to take an HSSP image copy with an application that does random processing instead of sequential, you can expect the application to promptly go into convulsions and die.

Chapter 6 DBRC Data Set Control

This chapter discusses DBRC's handling of data sets—logs, image copies, etc.—in RECON.

In DBRC terms, a log data set is an entry in a PRILOG record. In a batch environment, the PRILOG record contains a single data set entry. In an online environment, the PRILOG records can contain multiple data set entries, and a data set entry is added to the PRILOG record each time an archive is successfully performed. In a CICS-DL/1 environment, a data set entry is added each time a journal extent switch occurs (when the journal extent is subsequently archived, the data set entry will be updated to reflect the new copied data set information). In all cases, the data set entry could contain multiple volumes.

The forward recovery utilities only accept logs that have been archived. You cannot use OLDS directly. DBRC always looks in the PRILOG records when executing GENJCL.CA or GENJCL.RECOV. If you create an RLDS, DBRC uses this automatically. CICS-DL/1 and batch create PRILOG records.

Log Tape Management

DBRC is *not* a log tape management system. It does not stop you from reusing a log that is already recorded in the RECONs. Some customers have written native VSAM programs to interrogate the RECONs and see which logs are in use, subtract these logs from the pool, and generate a list of available logs. Other customers have developed programs based on the logs that are deleted from the RECONs as part of the regular cleanup.

The only data set control mechanism that DBRC provides, apart from the archive process, is the ability to predefine image copy and change accumulation data sets. DBRC cycles around and reuses the image copy and change accumulation data sets. See Chapter 5, “DBRC Registration” for a description of this procedure.

Generation Data Groups (GDGs)

DBRC fully supports the use of GDGs. They are recorded in RECON with their fully expanded names. See Chapter 5, “DBRC Registration” for a description of how to use them for image copy and change accumulation data sets.

GDGs can also be used for logs. No special action is required in DBRC to use them. The major thing to remember about GDGs is that they are normally updated at the end of the job (although you can choose to make this end of jobstep in an SMS environment), whereas DBRC is updated at the end of each job step.

If you ever generate multi-step jobs with DBRC (multiple steps doing the same thing, e.g., archives, change accumulations), DBRC will generate the same GDG numbers in each step by default. See Appendix C, “DBRC GENJCL Enhancements” for a description of the techniques required to solve this problem.

Log Information in RECON

DBRC stores information about batch logs, online logs, and archive output data sets. Batch logs are stored in PRILOG/SECLOG records. Online logs are recorded in PRIOLD and SECOLD records, and archive output is stored in a combination of PRILOG, SECLOG, PRISLD, and SECSLD records depending on your archive JCL setup. See Appendix B, “The Real Rules of Archive” for details of how archive works and what records are written to RECON.

In a log control environment, DBRC will simply record log information, but the associated LOGALL records will be empty—these only get entries written to them when you update a registered DBDS/area.

DBRC cleans up most of the records in RECON automatically—see "GENMAX Cycle—Log Cleanup in RECON" on page 45 for details. It is your responsibility, however, to get rid of unwanted—in DBRC speak INACTIVE—log records/log entries. The command to do this is DELETE.LOG INACTIVE.

DELETE.LOG INACTIVE

This command deletes log records (PRILOG, SECLOG, PRISLD, SECSLD) that have no LOGALL information (i.e., are no longer needed for recovery) and are older than the log retention period (LOGRET parameter on INIT.RECON). LOGALL information is cleared when ALLOCs are cleared (including RECOV, REORG records, etc.), which is done by cycling through the image copy GENMAX cycle. In a Log Control environment, this command will delete all log information from RECON, which is closed and older than LOGRET.

The RECONs must be large enough to hold log information back to the oldest image copy in RECON. If your RECONs are growing and the DELETE.LOG INACTIVE process appears to be ineffective, then you are probably suffering from one of the following problems:

- DELETE.LOG INACTIVE stops when it finds an open log. So do a LIST.LOG OPEN first. Close any logs that are not currently active (typically one that ran a long time ago when you were trying out DBRC and was not cleaned up properly) using NOTIFY.PRILOG, and then do DELETE.LOG INACTIVE. Details on how to close batch logs and online logs is given in the *DBRC Guide and Reference.* If you have problems closing log records in RECON, look carefully at the table in the manual showing the combination of parameters needed—I have frequently been caught by forgetting to code the SSID parameter, for instance.

- There are one or more databases that have not been image copied for a long time. If DELETE.LOG INACTIVE does not delete anything and there are no old OPEN logs in RECON, then do a LIST.LOG ALL and look at the first PRILOG/LOGALL combination listed.The LOGALL record will contain a list of the DBDSs, for which DBRC considers this log still holds valid recovery information, or in other words the DBDSs that have not been image copied frequently enough to cycle round GENMAX and trigger the ALLOC/LOGALL cleanup. Copy these DBDSs and your problems will go away.

- Too many generations of image copy defined in RECON (GENMAX).

  In this case, image copy more frequently or reduce the number of generations.

- You've forgotten to dump the message queues for ages—see below.

Never let the RECONs grow larger than 80 cylinders. This triggers a third level of VSAM index and impacts your performance. Most RECONs are smaller than 10 cylinders.

You should also consider using BMC Software's DATA ACCELERATOR Compression to compress the RECONs. This will reduce the number of I/Os for LISTs, GENJCLs, etc.

Continuous Operations

Prior to IMS V4, DELETE.LOG only deletes complete records, not entries within them. This has implications in continuous operations.

- The first problem is that prior to V4 you have to close down IMS before the PRILOG record fills up—the only way prior to V4 to get a new PRILOG record and hence room to write archive output details is to close IMS down and start it up again.

- The second problem is when you run IMS for four weeks, say, and you image copy weekly with a GENMAX of three. Now you close down IMS and run DELETE.LOG INACTIVE. Because the DELETE.LOG command deletes complete log records and not entries within them, the DELETE.LOG INACTIVE command achieves nothing as the log records are still required for their entries, which are more recent than the oldest image copy.

As from IMS V4, DELETE.LOG INACTIVE will delete *inactive* entries from log records and shuffle the remaining entries to the top of the record, hence creating more space—it calls this PRILOG compression. This means that you can run IMS continuously for months on end. IMS V4 will also try to compress these log records each time you run archive.

```
PRILOG
 START = 1997.274 08:47:55.2 -09:00     *    SSID=SYS3     VERSION=6.1
 STOP  = 1997.274 09:08:18.6 -09:00          #DSN=4
 GSGNAME=IMSGSG1
 FIRST RECORD ID= 0000000000000001       PRILOG TOKEN= 2
 EARLIEST CHECKPOINT = 1997.274 08:51:02.4 -09:00

 DSN=**** COMPRESSED DATA SET ****                        UNIT=
 START = 1997.274 08:47:55.2 -09:00       FIRST DS LSN= 0000000000000001
 STOP  = 1997.274 08:49:27.4 -09:00       LAST  DS LSN= 00000000000002DA
 FILE SEQ=0000    #VOLUMES=0000
```

Figure 22 Part of a COMPRESSED PRILOG Record

All inactive log data set entries in the current PRILOG (SECLOG, PRISLD, SECSLD) will be deleted automatically at archive time if the size of the PRILOG record exceeds 50% of the maximum RECON record size. DBRC considers a log entry to be INACTIVE if it is older than all the following criteria:

- log retention period
- oldest ALLOC on that log (in other words, you've image copied GENMAX times since that entry)
- earliest restart checkpoint for the online IMS system. The most common cause of this one is that you have forgotten to issue a /CHE SNAPQ for ages and IMS has to keep all the SLDS back to the last dump of the message queues in case of restart.

One customer was worried about the performance implications of this—did this mean that DBRC had to scan the whole RECON to find the oldest ALLOC each time they ran archive? What happens is that DBRC looks in the LOGALL record associated with the PRILOG, which provides it with a list of the DBDSs updated on that log. It then goes and looks for the first ALLOC for each of those DBDSs to determine the earliest ALLOC timestamp. This, in my opinion, could still be significant if you have thousands of databases.

Some DELETE.LOG Tests

- DELETE.LOG STARTIME deletes PRILOG and primary system log data set (PRISLD) records with STOP times of zero (even though some manuals say it will not). Some customers use this for cleaning up old records in RECON. However, there is a chance you will delete the open PRILOG and PRISLD records being used by the archive utility, or leave some orphan ALLOC records (i.e., ALLOC records pointing at a PRILOG that does not exist). A much safer method is to close the log using DFSULTR0 or NOTIFY.PRILOG. Image copy cleanup and DELETE.LOG INACTIVE will then get rid of the log record and any associated ALLOCs.

  One customer deleted some old OPEN logs and then discovered that they had 50 orphaned ALLOCs that they had to remove manually.

- When using DELETE.LOG STARTIME, you code the start timestamp of the record, and it will delete the whole record.

- When using DELETE.LOG TOTIME, you can code any 12-digit timestamp. It does not have to be the timestamp of a particular record. This command deletes any PRILOG, PRISLD, or LOGALL records whose starting timestamp is earlier than the TOTIME specified when the following conditions exist:

  — The STOPTIME is not zeroes.
  — The record is older than LOGRET (based on STARTIME).

List of Logs Deleted from RECON

IMS V1.3 used to produce a list of the logs deleted from RECON during the DELETE.LOG INACTIVE process. This list disappeared in IMS V2—it only indicated the number of deleted data sets. This causes problems for customers who post-process this list to determine which logs are reusable. The good news is that it is back again now—use the parameter LISTDL on INIT.RECON.

Size of PRISLD/PRILOG Record

If you are running IMS for any length of time, e.g., days or weeks or longer, then the PRILOG/PRISLD etc. records are going to get bigger and bigger as new entries are written to them each time you run archive. So you have to calculate the size of these records, in order to be able to select a sensible RECORDSIZE for the RECONs. The formula for calculating the size of a PRISLD/PRILOG record (this is for IMS V6—previous versions will be smaller, but it does no harm to over-allocate) is:

```
S = 112 + 120 D + 32 V
```

where *S* is the size of the PRISLD/PRILOG record in bytes, *D* is the number of SLDS/RLDS data sets created from archives for this execution of the online system, and *V* is the number of volumes for these SLDS/RLDS data sets.

For each unarchived OLDS, DBRC assumes that it will require 16 VOLSERs, which gives a size of 120 + 16 x 32 = 632 bytes. When you switch OLDS, DBRC wants enough room left in the record to archive this one and the one you have just opened. So work out the formula above based on the number of archives you expect to do between the oldest image copy and now, add 2 x 632 bytes and that gives you the maximum record size. For instance, with GENMAX = 3, image copy once per week, ARC = 1 archiving to a single cartridge each time and archiving 20 times per day gives:

```
112 + 120 x (21 x 20) +32 x (21 x 20) + 632 x 2 = 65216 bytes
```

REPRO will not copy a VSAM KSDS to a non-VSAM data set when the maximum record size is greater than 32,760 bytes. You have to back up the RECON to VSAM KSDS. When using this method of backup (rather than to non-VSAM data sets), you must use empty—IDCAMS DELETE/DEFINEd—KSDS. Otherwise your output will be merged with the previous backup copy.

For performance reasons, it is recommended you use SPANNED records and a CISIZE of 8K for the data and 2K for the index in RECON.

DBRC and Data Facility Hierarchical Storage Manager (DFHSM)

DFHSM does not have a DBRC interface. DBRC is not aware that a data set is taken away by DFHSM. This in itself does not matter as DFHSM migrates it back if it is required. The problem is that DBRC insists that it is on the same volume serial number (VOLSER) as the one from which it migrated because DBRC checks the VOLSER against what is recorded in RECON. Recalling to the same VOLSER can be forced via the exit in DFHSM. However, this assumes that there is still space available on that volume.

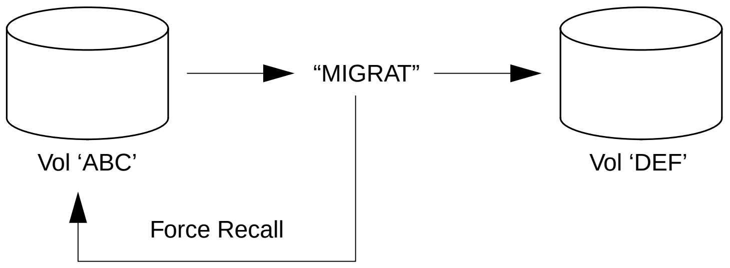


Figure 23 DFHSM Flow

DBRC caters for this with a parameter on the INIT.RECON and CHANGE.RECON commands: CATDS/NOCATDS. If you want DBRC to use the catalog to find data sets, remove VOLSER and UNIT information from the skeletal JCL and specify CATDS. DBRC still records volume and unit information in RECON and will check them if VOLSER and UNIT exist in the JCL.

This parameter is at the RECON level and applies to all logs, image copies, and change accumulations recorded in RECON. It has implications on JCL with disks and tapes as discussed below. It does not apply to DFSULTR0—always code all the information in the JCL for this utility.

Concatenated Tape and Disk

IMS allows you to log batch jobs to disk data sets (batch backout supports backout from disk). Many people use the combination of archiving their online logs to cartridges and using disk logs for batch processing. DBRC, however, generates JCL for concatenated data sets from one sample DD statement. So it is impossible to generate JCL with mixed devices using basic DBRC.

When you initialize RECON (or issue CHANGE.RECON) you place in the RECON header record default UNIT information for tape and DASD using the TAPEUNIT and DASDUNIT parameters.

DBRC identifies if a data set is on tape or DASD and stores that information in RECON. It does this for logs, image copy data sets, and change accumulation data sets. Every time you create one of these data sets, DBRC extracts a one-bit flag from MVS that indicates whether the data set is on tape or DASD. It then checks the RECON header record to see what you defined as the default tape and DASD units. So, if you have specified TAPEUNIT(3480), all data sets that are on tape will be stored in RECON with a UNIT of 3480.

The problems with this are:

- DBRC cannot tell the difference between 3420, 3480, and 3490, or between 3380 and 3390.

- You don’t always want to use UNIT=3480 for tapes anyway. It would be fine on image copy and change accumulation data sets, but it is far more efficient to use UNIT=AFF and save on tape units when reading concatenated log data sets.

Here are some solutions to the problem. The JCL you are trying to produce looks roughly like this:

```
//DFSULOG   DD DSN=.........,UNIT=AFF=DD3480
//         DD DSN=.........,UNIT=3390
//..
etc. etc.
```

There is a facility in DBRC that allows you to selectively generate any part of your skeletal JCL, depending on conditions you specify on the control card.

```
%DELETE (%LOGUINT EQ '3390')
//     UNIT=AFF=DD3480
%ENDDEL
```

Figure 24 Using DELETE Logic with LOGUNIT

The above example JCL only generates the UNIT information if the data set is not on DASD. Note, however, that you cannot tell the difference between different tapes or different disks. For disk, you could, of course, use a generic name, e.g., UNIT=SYSALLDA.

The select and delete logic is very powerful, but is not easy to understand at first sight. You are recommended to study the examples in my breathtakingly interesting manual—*GENJCL.USER Examples and Description*—and then to try some simple options first. Here is an example of generating concatenated tape and disk using %select logic:

```
%SELECT RLDS((%CAGRP),(FROM(%DSLLGTM)))
//DFSULOG  DD DSN=%LOGDSN,
%DELETE (%LOGUNIT NE 'CART')
//            VOL=(PRIVATE,,%LOGVSEQ,,SER=(%LOGVOLS)),
//            UNIT=AFF=DD3480,
%ENDDEL
%DELETE (%LOGUNIT NE 'SYSALLDA')
//         DD UNIT=SYSALLDA,VOL=SER=%LOGVOLS,
%ENDDEL
//                        DISP=OLD
%ENDSEL
```

Figure 25 Sample JCL for Concatenated Tape and Disk

However, if you are using CATDS, you do not want to generate the UNIT parameter on the disk data sets, but you do want to code UNIT=AFF on any tape data sets to reduce the number of tape drives required. Code a %DELETE, %ENDDEL pair to select UNIT=AFF only if the data set is not on disk.

```
%SELECT RLDS((%CAGRP),(FROM(%DSLLGTM)))
//DFSULOG DD DSN=%LOGDSN,
%DELETE (%LOGUNIT EQ‘SYSALLDA’)
//           VOL=(PRIVATE,,%LOGVSEQ,,SER=(%LOGVOLS)),
//           UNIT=AFF=DD3480,
%ENDDEL
//           DISP=OLD
%ENDSEL
```

Figure 26 Sample JCL for Concatenated Tape and Disk with CATDS

You may be wondering why I am using %DELETE instead of %SELECT. If you are in the middle of a %SELECT, %ENDSEL pair that you are already using to select the logs, you are not allowed to nest one %SELECT pair inside another.

Note: When writing this conditional JCL, make sure that you test it all possible ways, because it is dead simple to end up with invalid JCL—e.g., you forgot a comma or part of a DD statement. DBRC does not check your JCL for syntax—it generates anything you throw at it.

My JCL tends to be conceptual; if there are syntax errors, please don't call me to complain!

Batch Logging

Following are some suggestions regarding stacking batch logs, batch disk logging, and checkpoint and extended restart.

Stacking Batch Logs

Many customers now use disk logging in batch processing because it makes operations and JCL much easier. Problems start when you want to migrate the data sets from disk to tape. You cannot use DISP=MOD, because each log must be a separate data set so that you can recover to the end of it or back out from it.

The easiest way to migrate these data sets is to use DFHSM. You can use DFSUARC0 to migrate these data sets; it will update RECON, but you have to generate the JCL yourself.

Batch Disk Logging Considerations

When using disks for batch logging, consider the following:

- Block size

  If you make the block size large and the job has to perform many buffer flushes, then you will probably write many short records and waste a lot of disk space. Conversely, if the job only performs a few buffer flushes, the job requires larger blocks. The default chosen by IMS is 1K.

- Out of space

  Miscalculation of sizes can lead to B37 abends.

- BKO=Y

  This parameter, designed for the block-level sharing user, performs dynamic backout in the event of certain, but *not all*, batch failures. It works by writing fixed-length records to the log and padding the blocks where necessary. It is not recommended due to its high DASD usage. Also, it does *not* work in *all* circumstances.

- IDRC

  IBM does not recommend the use of hardware compression on IMS batch logs. This is because it makes batch backout run like a dog—it has to expand each record to find the backwards pointer to the previous one.

Use of BMC Software's BATCH CONTROL FACILITY avoids all these problems.

Checkpoint and Extended Restart

Using CHKPT/XRST is recommended in long-running batch jobs, and is essential in n-way sharing environments. Backout is to the last checkpoint by default in non-sharing environments, and always in sharing environments.

Note: It is your responsibility to start the job from the correct checkpoint. IMS allows you to restart from the wrong one. BMC Software's APPLICATION RESTART CONTROL product (which includes batch DL/1 restart) solves this problem. It will also automatically insert checkpoint/restart into programs for you without having to write any application code yourself.

Chapter 7 Bad Data Sets

If you run a utility and there are problems with the output data sets, then DBRC will be aware that the utility was unsuccessful and the normal action is to rerun the utility. If, on the other hand, you are running a utility where you have problems with reading one of the input data sets (e.g., image copy data set in recovery), then you will discover that these read I/Os are not notified to DBRC. You have to tell DBRC that the data sets are bad.

When DBRC generates JCL, it looks at the PRILOG records and primary image copy records. To force it to use secondary data sets, you have to tell it that the primary is broken.

Bad Image Copy Data Set

If an image copy is unreadable, DBRC does not automatically select the secondary image copy. You must flag the primary image copy as being broken and regenerate the JCL. DBRC then selects the secondary. The command to do this is CHANGE.IC... INVALID—see my *GENJCL.USER Examples and Description* manual for an example. If you have been really sensible and have bought BMC Software's RECOVERY PLUS, then you don't need to do anything to DBRC—just use the SIC parameter (aren't we nice to you!?)

If you do not have a secondary image copy, DBRC selects the previous generation, the necessary logs, and change accumulation, but if there has been a reorganization since the last good image copy, then DBRC cannot generate any JCL for you.

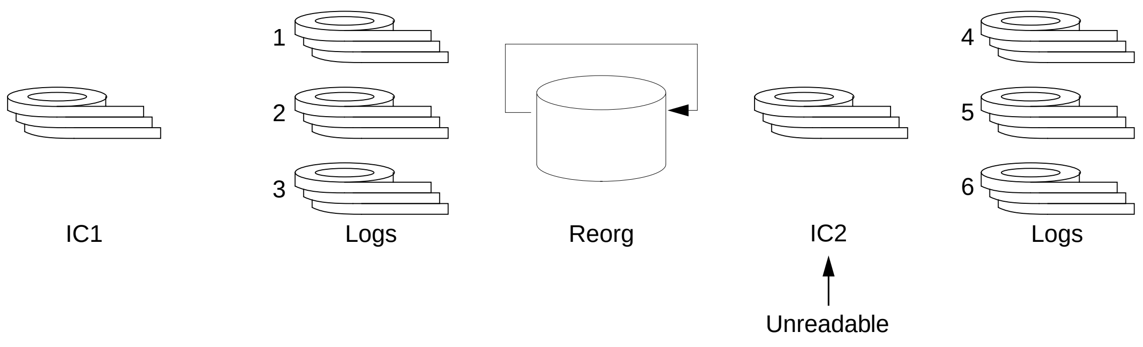


Figure 27 DBRC and Unreadable Image Copies

Your possible actions are:

1. If the broken database is an index, use BMC Software's SECONDARY INDEX UTILITY/EP to rebuild it.

   OR

2. If you still have the unload data set, rerun the reload outside DBRC (you run this outside DBRC as you don't want RECON to have yet another REORG record to get confused by).

   NOTIFY this to DBRC as a timestamp recovery to IC2 (i.e. RCVTIME=time of IC2 and RUNTIME=CURRENT), followed by a GENJCL.RECOV with the USEDBDS parameter. This will apply logs 4, 5, 6 on top of the database.

   This is horrendous and not recommended for the faint-hearted. I tried it once and got it to work after much blood, sweat, and tears. Avoid if you can.

   OR

3. Run a timestamp recovery to the end of log 3. Now, reorganize the database (outside DBRC) exactly as the previous reorg was run. Then continue as in the second course of action, listed above. This, of course, assumes that reorg runs exactly the same way twice, which I frankly would not count on.

The moral here is to use dual image copy output after a reorg!

Bad Change Accumulation Data Set

There is no support in IMS for dual change accumulation output data sets. If you get an error here, mark the data set as bad using the CHANGE.CA ... INVALID command. The next time you run change accumulation, DBRC will select the appropriate input data sets.

If you have been sensible once again and bought BMC Software's CHANGE ACCUMULATION PLUS, you will find that it supports dual output. Unfortunately, DBRC only allows us to register the primary copy in RECON. If this later proves to be unreadable, then use a CHANGE.CA command to point at the secondary copy and rerun your job—again, there is an example in my *GENJCL.USER Examples and Description* manual.

How to Use a SLDS in Place of a Bad RLDS

You cannot use DFSULTR0 to repair an RLDS because DFSULTR0 checks sequence numbers that are not contiguous in an RLDS.

If a write error occurs on an RLDS during archive, rerun the archive. If a read error occurs on an RLDS and there is a secondary RLDS available, flag the primary as being in error by using the CHANGE.PRILOG ... ERROR command. Regenerate the JCL and DBRC will select the secondary RLDS (SECLOG record in RECON).

Note: BMC Software's RECOVERY PLUS product (the one you licensed two pages ago) has a special parameter SLOG, which selects the secondary log(s) without having to issue DBRC commands to flag the primary logs(s) as ERROR. Will this generosity never end?

If a read error occurs on an RLDS and there is no secondary RLDS available, use the SLDS instead. DBRC does not automatically select the SLDS instead of the RLDS (because it looks at PRILOG records for GENJCL.CA and GENJCL.RECOV, not PRISLD records). If you want to update RECON to reflect this (or you are in DBRC Recovery or Share Control where you must update RECON), perform one of the following steps:

- Manually update the PRILOG record in RECON (with DSN and VOLSER information) to point at the SLDS.

  ```
  CHANGE.PRILOG RLDS STARTIME(...) DSN(new dsname) OLDVOL(oldvol) -
  NEWVOL(newvol) DSSTART(...)
  ```

  OR

- Copy the SLDS to a new data set with the same name and VOLSER as the bad RLDS using IEGBENER or DFSUARC0. In this case, no update of RECON is necessary.

  If the RLDS has been produced by archiving an OLDS, use DFSUARC0 with PARM='DBRC=N' (which is possible even if you have DBRC=FORCE) and use the SLDS as input. This will run outside DBRC control.

Warning! I have tried running this with DBRC turned on, and the resulting updates in RECON are unpredictable because DBRC thinks you are trying to archive the SLDS as a batch log. However, the PRILOG/PRISLD records have multiple entries (online record types) and DBRC gets very confused.

A Clever Archiving Technique

The following technique is used by some customers to give them secondary SLDS and secondary RLDS without having to create both data sets.

Archive output is:

- primary SLDS — PRISLD record in RECON
- secondary SLDS — SECSLD record in RECON (will be used if PRISLD marked in ERROR)
- primary RLDS — PRILOG record in RECON
- no secondary RLDS

  In this case, DBRC still builds a SECLOG record that points at the secondary SLDS data set. This means both the SECLOG and SECSLD records contain the same information and are used if PRILOG is marked in ERROR.

If you use this technique, you must ensure that the PRILOG and SECSLD are on the same device type, or use one of the techniques described in “Concatenated Tape and Disk” on page 62 to handle mixed device types.

Chapter 8 Batch Backout, Signon, and Authorization

I have come across more errors in batch backout with DBRC than with any other utility. Whereas the interface with DBRC on the forward recovery utilities and in an online environment is good, in the case of batch and batch backout, it definitely leaves a lot to be desired. You are recommended to use BMPs or buy BMC Software's BATCH CONTROL FACILITY.

BMPs have a lot going for them—single log, automatic archive, dynamic backout, easy restart with BMC Software's APPLICATION RESTART CONTROL, etc. They have an overhead compared with batch, but I know many customers who consider this worthwhile in light of the operational advantages. DBCTL also makes BMPs available to CICS users, who have had to suffer the abysmal performance of CICS shared batch or the operational complexities of block-level sharing for many years.

I also appreciate, however, that conversion of all batch work to BMPs for some customers is neither desirable or feasible. In my opinion, batch work will be with us for many years yet!

The purpose of this chapter is to provide some detail of how the IMS batch backout utility interfaces with DBRC. It also provides guidelines on the use of the utility, and describes the basic principles of SIGNON and AUTHORIZATION.

Background

All changes made to IMS databases by an IMS DB/DC or DBCTL subsystem are logged to its log data sets. Logging is also available to the IMS DB subsystems. The data logged to these files differs, depending on whether the databases involved are full function or Fast Path databases. (Fast Path databases cannot be accessed via batch.)

In the case of full-function databases, the information held on the log is the image of the database record both before and after the update (the exception to this is NONRECOVerable databases—see “RECOVABL/NONRECOV” on page 42 for further details). The data logged for Fast Path databases only includes the *after* image.

Before and *after* images are held for different purposes. The *before* images are used for backout purposes, whereas the *after* images are used for database recovery. Backout processing cannot be carried out against Fast Path databases—they are recovered using the forward recovery utility.

This chapter is only concerned with the use of the batch backout utility with full-function databases.

Batch Backout Utility

Although details of this utility can be found in various IMS manuals, some repetition is required in this book for the sake of completeness.

The utility is used to recover databases to a point before a program was initiated, or to a checkpoint or sync point. It backs out all updates for all DL/1 data sets performed in that time period. It is run as a normal IMS batch job and uses the program specification block (PSB) of the program whose updates are to be backed out.

The program does not carry out physical replacement, but uses DL/1 space management to free or restore space. This means that a segment deleted by the failing program and reinserted by batch backout may not be in its original position in the database. The output log must be kept for input to any subsequent recovery cycle, since it and the original logged change refer to different physical disk addresses. Both are required for recovery. DBRC records both logs and includes both in any subsequent recovery.

DBRC Processing Prior to Batch Backout

This section describes the DBRC processing carried out by IMS subsystems (batch and online) that update databases. These details provide some background information on the recording carried out by DBRC prior to and during a subsystem failure that results in the requirement to execute the batch backout utility.

Subsystem Signon and Signoff

Subsystem signon and signoff are only relevant to DBRC Share Control environments. The Recovery Control environment does not perform these functions. The only authorization check that Recovery Control performs is verifying the database data set name is the same as in RECON. If the names do not match and if you have specified FORCER (force registration of all databases), authorization will fail. Otherwise DBRC will just issue warning messages saying the database is not registered.

Signon

When an IMS batch, utility, or online subsystem is initialized, IMS registers it with DBRC by performing SIGNON NORMAL. This creates a SUBSYS record in RECON. This record contains details of the subsystem identifier (SSID), which must be unique, the subsystem type (batch or online), the start time of the log, a list of authorized databases (added later), and some flags. The SSID for batch and utility subsystems is the job name. The SSID for IMS online (non-XRF) subsystems is the IMSID parameter, and for CICS-DL/1 and XRF IMS online subsystems it is the virtual telecommunications access method (VTAM) APPLid.

Authorization

Part of the signon process for batch and utility subsystems is known as authorization. For online subsystems, this authorization process takes place at PSB scheduling time for databases that have not already been authorized to that subsystem. The basic purpose of authorization is as follows:

- to prevent access to a database when
  - — the database needs a utility operation
  - — the database is already authorized to another subsystem at a conflicting level

- to allow access to a database when
  - the database is not already authorized to another subsystem
  - the database allows sharing (and no conflict detected)
- to keep a record of current authorization levels held for subsystems in relation to databases

Authorization will be successful only if all databases specified in the request can be authorized, even though it does not require all of the databases to be registered with DBRC (unless you have specified FORCER in RECON). In batch, the authorization is based on the PROCOPT; in online systems, it is based on the ACCESS parameter on the DATABASE macro or the last /STA command; and for utilities, it is hard-coded. The four possibilities are READ-GO, READ, UPDATE, or EXCLUSIVE.

When authorization is complete, the SUBSYS record, as described earlier, is updated with the names of authorized databases. DB (database) records in RECON are also updated with details of the subsystem and its authorization levels.

If authorization fails, the action taken by IMS depends upon whether the subsystem is an online or batch subsystem. For batch and non-message driven BMPs, the subsystem is abended with a U0047 abend. For online subsystems, the DBD is stopped and the input message is requeued. Batch backout is not required for such failures as no databases will have been updated. For batch subsystems, the SUBSYS record is also removed from RECON.

```
SSY
 SSID=SYS3      LOG START=1997.274 09:17:31.2 -09:00
 SSTYPE=ONLINE  ABNORMAL TERM=OFF  RECOVERY STARTED=NO   BACKUP=NO
 TRACKED=NO     TRACKER TERM=OFF   SHARING COVERED DBS=NO
 IRLMID=**NULL**  IRLM STATUS=NORMAL        GSGNAME=IMSGSG1

   AUTHORIZED DATA BASES/AREAS=1      VERSION=6.1
                                                       ENCODED
     -DBD-        -AREA-   -LEVEL-  -ACCESS INTENT-  -STATE-
     HDAMOSAM     **NULL**    1        UPDATE           6
```

Figure 28 Sample SUBSYS Record

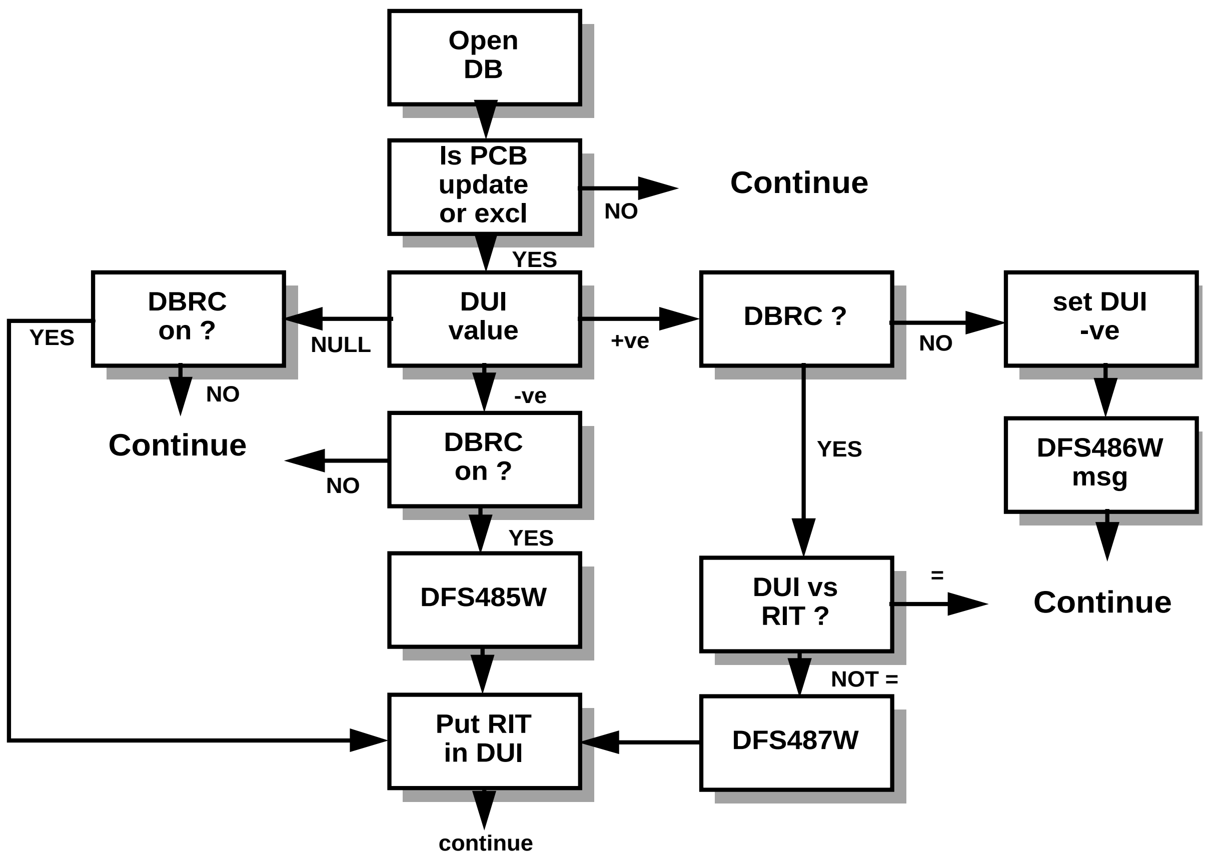
```
DB
 DBD=HDAMOSAM              IRLMID=*NULL        DMB#=2        TYPE=IMS

 SHARE LEVEL=3              GSGNAME=**NULL**     USID=0000000006
 AUTHORIZED USID=0000000006  RECEIVE USID=0000000006 HARD USID=0000000006
 RECEIVE NEEDED USID=0000000000
 FLAGS:                                COUNTERS:
   BACKOUT NEEDED        =OFF            RECOVERY NEEDED COUNT   =0
   READ ONLY             =OFF            IMAGE COPY NEEDED COUNT =0
   PROHIBIT AUTHORIZATION=OFF            AUTHORIZED SUBSYSTEMS   =1
   RECOVERABLE           =YES            HELD AUTHORIZATION STATE=6
                                         EEQE COUNT              =0
   TRACKING SUSPENDED    =NO             RECEIVE REQUIRED COUNT  =0
   OFR REQUIRED          =NO

   ASSOCIATED SUBSYSTEM INFORMATION:
                                        ENCODED
           -SSID-   -ACCESS INTENT-  -STATE-  -SS ROLE-
            SYS3        UPDATE          6      ACTIVE
```

Figure 29 Sample DB Record

Authorization is retained until the subsystem terminates or until the database is taken offline via /DBR. Authorization can also be changed via the /STA DB ACCESS= and /DBD commands.

Note: For Fast Path DEDBs, authorization is requested at the first DL/1 call (because of the area concept). An authorization failure results in an FH status code.

DUI Processing



-ve = negative
+ve = positive

Figure 30 DUI Processing

The DUI is a 4-byte field kept in either the database data set or its entry in the VSAM catalog and is derived via hashing from the RECON initialization timestamp (RIT), which is stored in the RECON header record. DUI processing is performed at DB open time for DL/I and DEDB databases if the subsystem requires UPDATE or EXCLUSIVE access.

Note: The user will receive a DB open logic error if the DB is registered with DBRC but not all the database data sets are known to DBRC.

Warning! This is a really clever piece of code that tells you when you are doing it wrong—turning DBRC on and off or using the wrong set of RECONs, but these are only warning messages! To avoid this, use the options of FORCE and FORCER.

Log Registration

Following successful signon and authorization, subsystems continue through log registration.

DBRC demands that all batch systems with a PROCOPT>G have a log. You cannot DUMMY it out. This does not apply to PROCOPT=L(S).

Note: This is based on the PROCOPT, *not* on whether the job actually updates.

BMC Software’s BATCH CONTROL FACILITY contains a feature called Exchange Log Device Type, which enables you to implement logging in batch jobs (even on jobs with DD DUMMY specified) *without* having to change your JCL. It will also allow you to run batch update jobs with no log with DBRC turned on.

DBRC recognizes that most utilities run without a log and allows this. (Prefix update can run with a log. It then behaves like a normal batch job. However, this is not recommended, as it only makes the job slower.) Actually if you really want to make prefix update zip along, then BMC Software’s PREFIX RESOLUTION PLUS is the chap for you—see Chapter 12, “BMC Software and DBRC”.

When the primary system log is opened, either by a DL/1 batch subsystem or an online subsystem, DBRC creates a PRILOG record in RECON. If dual logging is used, a SECLOG record is also created (for online IMS systems this record is created when you run the first archive, if your DD statements in the archive job specify a secondary SLDS and/or a secondary RLDS). These records contain the data set name, volume serial number, and OPEN timestamp for the log data set(s). In addition, a LOGALL record with the same timestamp as the PRILOG is created. This is used to hold a list of registered database data sets with change and/or I/O error records on the log. The list is empty when the LOGALL record is created.

```
PRILOG
 START = 1997.274 09:08:59.4 -09:00     *    SSID=BATCH1   VERSION=6.1
 STOP  = 1997.274 09:09:05.8 -09:00          #DSN=1
 GSGNAME=IMSGSG1
 FIRST RECORD ID= 0000000000000001       PRILOG TOKEN= 3

 DSN=BATCH1.UPDATEF.LOG                                  UNIT=SYSDA
 START = 1997.274 09:08:59.4 -09:00       FIRST DS LSN= 0000000000000001
 STOP  = 1997.274 09:09:05.8 -09:00       LAST  DS LSN= 000000000000009C
 FILE SEQ=0001    #VOLUMES=0001

  VOLSER=000000 STOPTIME = 1997.274 09:09:05.8 -09:00
    CKPTCT=0    CHKPT ID = 0000.000 00:00:00.0 +00:00

LOGALL
 START   = 1997.274 09:08:59.4 -09:00     *
 DBDS ALLOC=3                         -DBD-    -DDN-    -ALLOC-
                                      DHVNTZ02 HIDAM     1
                                      DXVNTZ02 XDLBT04I  1
                                      DIVNTZ02 DBHVSAM1  1
```

Figure 31 Sample PRILOG and LOGALL Records

Note: The * indicates that the timestamp is part of the key.

Allocation Registration

When a DL/1 batch or IMS DB/DC or CICS-DL/1 online subsystem first updates a database data set and the first change record is written to the log, DBRC is invoked. The exit invoked is called the allocation exit.

Note: For Fast Path DEDBs, ALLOC happens at area open time, unless the PROCOPT is G or GO, when it is not required.

If the database data set is registered, an ALLOC record containing the timestamp of the update and the start time of the PRILOG record is created. Additionally, the database data set is added to the list of database data sets in the LOGALL record.

If the database is closed using the /DBR command, or the CICS equivalent, DBRC records the deallocation timestamp in the ALLOC record (/DBD, /STA DB ACCESS=RD, and /STA DB ACCESS=RO also put a DEALLOC time in RECON).

```
ALLOC
ALLOC   =1997.274 09:14:27.6 -09:00     *  ALLOC LRID =0000000000000000
DSSN=0000000001 USID=0000000002 START = 1997.274 09:12:08.5 -09:00
```

Figure 32 Sample ALLOC Record

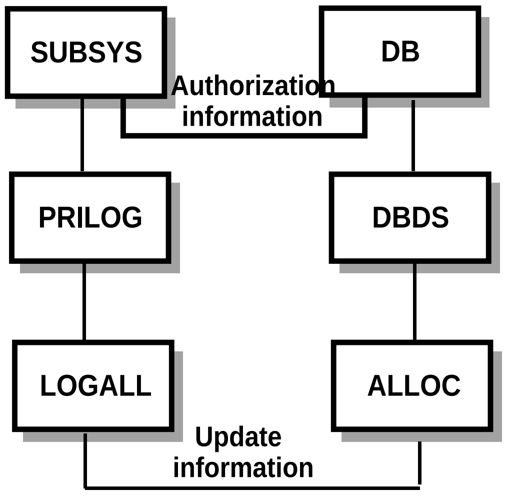
Batch jobs do not write DEALLOC timestamps. If a batch job/job step terminates normally, then the STOP time in the PRILOG record will be filled in with the end of job step timestamp and DBRC considers the databases to be "offline." The significance of DEALLOC or end of log data set is that DBRC will allow you to recover to a point where the database is "offline."

Note: A STOP time of zeroes in a log record means that the job step is running or the job has abended.

Signoff

When an IMS subsystem terminates, DBRC performs SUBSYSTEM SIGNOFF. This process differs, depending upon whether the subsystem terminates normally or abnormally.

Record Structure in RECON



The figure shows the records that are built/updated in RECON to show that a job is running (SUBSYS), databases are authorized and updates have taken place. It is the presence of this record structure in RECON that shows DBRC that cleanup actions are required—e.g., batch backout for failing batch updaters. The BACKOUT NEEDED flag is only turned on for online problems—dynamic backout failure etc.—it is not turned on for batch failures.

Normal Termination

On successful completion of the batch subsystem, utility, or online subsystem, IMS calls DBRC to perform SIGNOFF NORMAL, which carries out the following:

- Database(s) are unauthorized as follows:
  - — Database name(s) are removed from the SUBSYS record.
  - — The subsystem identification is removed from database record(s).
  - — The held state is recalculated (or reset to zero).

- SUBSYS record is deleted.

Abnormal Termination

If the subsystem terminates abnormally, the action taken depends on whether or not (E)STAE is in control.

Abnormal Termination — (E)STAE in Control

If the subsystem terminates abnormally and (E)STAE is in control, IMS calls DBRC to perform a SIGNOFF ABNORMAL. The action taken depends on the type of subsystem.

Batch Subsystem

- Registered databases that have not been updated by this subsystem are unauthorized (see earlier). If no databases have been updated by the subsystem (or no updates have taken place since the last checkpoint), the unauthorization process deletes the SUBSYS record and the following message is issued:

  ```
  DFS036I BATCH BACKOUT NOT REQUIRED FOR job name
  ```

  and the batch job can be resubmitted or restarted.

- Registered databases that have been updated by this subsystem remain authorized. The SUBSYS record is flagged as ABNORMAL TERM = ON and kept in RECON (not deleted).

  Note: Databases updated by a subsystem are databases to which changes actually took place and for which ALLOC and LOGALL records were added to RECON.

  In this case, the following message is issued:

  ```
  DFS036I BATCH BACKOUT IS REQUIRED FOR job name
  ```

  Before rerunning or restarting the subsystem, run the batch backout utility or an alternative backout method. See “Alternative to Backout” on page 85 for details of alternative methods.

Non-Registered Databases

DBRC recognizes if non-registered databases were updated using a subsystem and retains the SUBSYS record to enforce batch backout. This enables users to get used to DBRC’s way of working prior to registering databases. However, this means that in a testing environment, jobs have to be properly cleaned up. Simply renaming the job and rerunning it will leave SUBSYS records and possible unclosed PRILOGs in RECON.

Online Subsystem

Exactly the same action takes place for online systems as for batch subsystems, except that message DFS036I is not displayed. The correct user action is to always use emergency restart.

If emergency restart cannot be performed, an unplanned cold start (/ERE COLDSYS or COLDCOMM or COLDBASE) must be performed after running batch backout for all online programs executing at the time of the failure. See later in this chapter for additional information.

Utility Subsystem or an Initial Load Program

The registered databases are unauthorized and the SUBSYS record is deleted from RECON. The subsystem can simply be resubmitted.

Abnormal Termination — (E)STAE Not in Control

If the subsystem terminates abnormally and (E)STAE is not in control (e.g. CPU, MVS, or (E)STAE failure), DBRC cannot perform a SIGNOFF ABNORMAL. The SUBSYS record and any database authorizations remain in RECON, and the SUBSYS record is not updated to reflect the subsystem failure. The action taken by the user depends on the type of subsystem.

Batch Subsystem

- If registered databases were updated by this subsystem, close the log and use the batch backout utility or an alternative backout method. (See "Alternative to Backout" on page 85 for details of such a method.) Now resubmit or restart the batch job.

- If registered databases have not been updated by this subsystem, either run the batch backout utility (if a log was created) or use the CHANGE.SUBSYS SSID(ssname) ABNORMAL command. Note that batch backout will receive the following message:

```
DFS041I DBRC SIGNON REQUEST RC=24
```

  Batch backout will usually continue. If, however, it later fails because of a database I/O error, use the CHANGE.SUBSYS command to mark the SUBSYS record as ABNORMAL and the Database Recovery utility to recover the database(s). See Chapter 11, "Recovery" for additional information. Complete the backout and then resubmit or restart the batch job.

Non-Registered Databases

If the job only uses non-registered databases, then issuing a CHANGE.SUBSYS ABNORMAL command will not delete the SUBSYS record, because DBRC maintains a bit in the LOGALL record to indicate that a non-registered database(s) has been updated, and handles the CHANGE.SUBSYS ABNORMAL as if you had registered databases—i.e. retains the SUBSYS if backout is needed.

Warning! The CHANGE.SUBSYS ABNORMAL command cannot tell the difference between a job that is running and one that terminated abnormally with (E)STAE not in control. *Never* issue this command against a subsystem which is currently running—only ever use it when you are sure the subsystem has failed, e.g. use it as the first step of your restart job.

Automating Backout with GENJCL.USER

Log closing and batch backout for batch subsystems can be partially automated using the GENJCL.USER facilities of DBRC, combined with the %SELECT and %DELETE commands—see my *GENJCL.USER Examples and Description* book if you are a masochist. The problems can be totally solved by BMC Software's BATCH CONTROL FACILITY.

Online Subsystem

The correct procedure is always to use emergency restart. IMS requires you to flag the SUBSYS record as abnormally terminated prior to /ERE (using CHANGE.SUBSYS SSID(ssname) ABNORMAL). If you forget to do this, IMS restart will hang waiting for another command—to fix this, you have to use the /ERE OVERRIDE command, which tells DBRC that you know what you are doing (you hope!) and it will ignore the fact that there is already a SUBSYS record in RECON for this IMS system you are restarting.

/ERE will usually continue. If, however, it fails because of a database I/O error, it will be necessary to use the CHANGE.SUBSYS command to mark the SUBSYS record as ABNORMAL and the Database Recovery utility to recover the database(s).

Utility Subsystem or an Initial Load Program

The CHANGE.SUBSYS SSID(ssname) ABNORMAL command must be used before resubmitting these jobs.

Warning! As stated above, DBRC cannot tell the difference between a job that is running and a job that failed abnormally without (E)STAE (log STOP time of zeroes and no flags set). Never issue CHANGE.SUBSYS ABNORMAL commands for jobs that are running. One customer had a problem with this—one MVS image went bang. Unfortunately, they then issued CHANGE.SUBSYS ABNORMAL commands for all the subsystems they found in RECON, which of course included all the jobs currently running on the second MVS image!

Batch Backout and ERE Subsystem Signon

The Batch Backout utility and emergency restart both signon in a special way (SIGNON RECOVERY START) by using the SUBSYS record of the failed subsystem. On successful completion of the batch backout utility or emergency restart's backout processing, DBRC deletes the SUBSYS record. This is known as SIGNON RECOVERY END. Emergency restart performs a fresh SIGNON NORMAL to register the online system in the RECON.

People always ask me whether this means that the backout job for a failed batch job has to run with the same job name as the failed job—answer is NO. What happens is that backout reads the input log, finds the job name, looks in RECON for a SUBSYS with this name (which it will find if the job failed, together with all the authorization information) and uses this SUBSYS and authorization to run the backout job.

If batch backout is run and a SUBSYS record does not exist, batch backout fails (U0041) and the following message is issued:

```
DFS041I DBRC SIGNON REQUEST RC=28
```

In this case, batch backout is not required. Backing out a normally terminated job (i.e., there is no SUBSYS record in RECON) is covered later in this chapter.

Now, when you are running backout for an online problem, e.g., dynamic backout, it is quite possible that IMS is up and running, so in this case backout signs on in the normal way and creates its own SUBSYS record.

Batch Backout Normal Termination

When batch backout terminates normally, the action taken depends on whether the input log was created by a batch or online subsystem.

- Batch input log
  - uses SUBSYS record of failed batch job
  - all databases are unauthorized
  - the SUBSYS record is deleted
- Online input log
  - signs on with new SUBSYS record
  - the backout needed counter(s) is decremented
  - if the counter(s) is zero, the backout needed flag(s) is reset

    The above activities are turned on by online backout failures.
- The SUBSYS record for the backout job is deleted.

Batch Backout Abnormal Termination

When batch backout terminates abnormally, all RECON records remain unchanged so backout can be rerun. You *must* keep all the logs from *all* the executions as they can contain database change records.

You rerun backout using the *original* closed log from the first failed job.

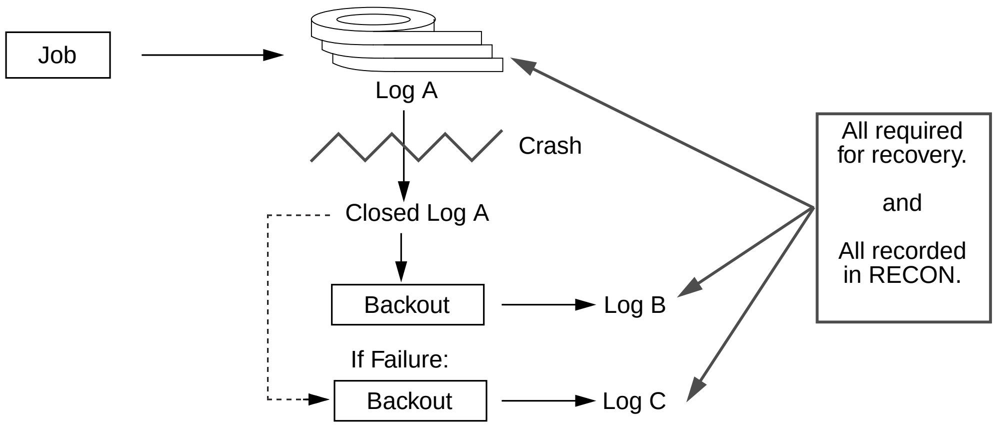


Figure 33 Batch Backout After Abnormal Termination Flow

Warning! Do not do what one customer did—backout failed, so they reran it with Log B as input (not Log A). This didn't work, so then they ran with Log C as input etc., etc. Then they rang me up complaining that it didn't work! I quietly and rationally explained that they were perhaps not doing it quite right and told them to try Log A—hey presto!

Warning! Batch backout logs don't contain before images—therefore, you can't backout a backout.

Alternative to Backout

Although it is recommended that batch backout is always used to perform backout of failed subsystems, there may be cases where it is not possible to run batch backout (for example, the input log is damaged). My favorite error here was a customer that ran a job with DISP=(CATLG,DELETE) on a disk batch log by mistake and of course the job abended—oh what fun!

It is also possible that some installations prefer not to log batch updating subsystems. Two alternative methods of performing backout are as follows:

First Method

This method uses timestamp recoveries.

Step 1 Use the CHANGE.DB DBD(dbdname) NOAUTH command to prohibit further authorization to all necessary databases. This step is optional.

Step 2 Use the CHANGE.SUBSYS SSID (ssidname) ABNORMAL command to unauthorize any non-updated databases. Note that this will not be required if (E)STAE has taken effect.

Step 3 Use the CHANGE.SUBSYS SSID(ssidname) STARTRCV command to flag the SUBSYS record as Recovery Started.

Step 4 Run timestamp recoveries for all necessary databases to restore these databases back to their status at the beginning of the subsystem that failed. It is vital that the correct databases and timestamps are selected.

See “Timestamp Recovery” on page 124 for an example of automating timestamp recovery of a group of databases back to the last image copy.

If you have been inspired enough to buy BMC Software’s RECOVERY MANAGER for IMS, then you will find that this product will find the timestamps for you and build all the JCL to recover a group of databases, which leaves you with a lot more time in life for important things like playing golf.

Step 5 Use the CHANGE.SUBSYS SSID(ssidname) ENDRECOV command to flag the SUBSYS record as RECOVERY ENDED.

Step 6 Use the DELETE.SUBSYS SSID (ssidname) command to delete the SUBSYS record.

Step 7 Use the CHANGE.DB DBD(dbdname) AUTH command to reset the prohibit authorization flags for all necessary databases if you used NOAUTH above.

Step 8 The failed subsystem can be resubmitted.

Note: It is your responsibility to ensure that *all* the necessary databases, indexes, operating system (OS) files, etc., have been recovered to the correct point in time.

Second Method

This method deletes all records from RECON for the failed job and uses full forward recoveries.

Step 1 Use the CHANGE.DB DBD(dbdname) NOAUTH command to prohibit further authorization to databases. This step is optional.

Step 2 Use CHANGE.SUBSYS SSID(ssname) ABNORMAL to tell DBRC that the subsystem has fallen over and to unauthorize any non-updated databases.

Step 3 Use DELETE.ALLOC DBD(..) DDN(..) RECTIME(..) commands to delete all the allocation records for the failed job.

Step 4 Run full recoveries for all necessary databases to restore them to their status at the beginning of the subsystem failure. It is vital that the correct databases are selected. DBRC does not include the log from the failed job because it does not believe there are any updates on it.

Step 5 Use CHANGE.SUBSYS SSID(ssidname) NORMAL, CHANGE.SUBSYS SSID(ssidname) ABNORMAL commands to cleanup the SUBSYS record and the authorizations. DBRC now believes all the databases are non-update because you have removed the ALLOC records, and will delete the SUBSYS record.

Step 6 Use the CHANGE.DB DBD(dbdname) AUTH command to reset the Prohibit Authorization flags for all necessary databases.

Step 7 The failed subsystem can be resubmitted.

Note: This procedure should be used only if you can automate it. An example follows along with an explanation of the %SELECT, %DELETE logic.

```
//CLEAN  EXEC PGM=DSPURX00
//SYSPRINT DD SYSOUT=A
//JCLPDS DD DSN=IMSVS.JCLPDS,DISP=SHR
//JCLOUT DD SYSOUT=(A,INTRDR)
//SYSIN DD *
CHANGE.SUBSYS (%SSID,ABNORMAL)
/*********************************************************************/
/* The SUBSYS record must be flagged as abnormally terminated to      */
/* allow the Recoveries to run                                        */
/* This next part  runs DBRC to select the most recent log (PRILOG)   */
/* for the job that failed.                                          */
/* Selecting information from RECON sets values for DBRC keywords -   */
/* which keywords are set are detailed in the Reference manual        */
/* In this example, selecting a PRILOG record sets:                   */
/* Keyword  %LOGSTIM to the log START TIME                            */
/* Keyword  %LOGETIM to the log STOP  TIME (zeroes if not closed)     */
/*********************************************************************/
%SELECT RLDS(%SSID,LAST)
%ENDSEL
/*********************************************************************/
/* Now you can use the values set in the keywords to select the       */
/* ALLOC records associated with this log.                           */
/* There will be an ALLOC record for each updated DBDS                */
/*********************************************************************/
%SELECT ALLOC(PRILOG,%LOGSTIM)
    /*DELETE THE ALLOC RECORDS                                        */
    DELETE.ALLOC DBD(%DBNAME) -
    DDN(%DBDDN) -
    RECTIME(%ALLTIME)
/*********************************************************************/
/* %ALLTIME was set by the %SELECT ALLOC                              */
/* The DELETE.ALLOC will be repeated for each ALLOC selected          */
/* Now the ALLOCs have gone, generate the Recoveries                  */
/* These can be FULL forward recoveries - DBRC will not include       */
/* the log from the failed job because it has no updates (ALLOCs)     */
/*********************************************************************/
    GENJCL.RECOV DBD(%DBNAME) DDN(%DBDDN)
%ENDSEL
/*********************************************************************/
/* You have now recovered all the updated DBs.                        */
/* You still need to cleanup the SUBSYS record and the authorizations  */
/* To do this, you have to trigger the ABNORMAL (ESTAE) processing    */
/* So turn the ABNORMAL flag off and then turn it on again to trigger  */
/* the cleanup                                                        */
/* This will delete the SUBSYS record, because there are no ALLOCs    */
/* and hence no updated databases to worry about                      */
/*********************************************************************/
CHANGE.SUBSYS (%job name) NORMAL
CHANGE.SUBSYS (%job name) ABNORMAL
/* Delete the PRILOG record from RECON if you wish                    */
DELETE.PRILOG STARTIME(%LOGSTIM)
```

Figure 34 Automatic Cleanup of a Failed Batch Job

Backing Out a Normally Completed Job

In a non block-level-sharing environment, you can use DBRC=C on the batch backout utility to backout a completed job. Actually, the correct definition is a non IMS resource lock manager (IRLM) environment, but it is not recommended that you use the IRLM unless you are using block-level sharing. Use DBRC=C with caution, because DBRC does *not* check to see if there are subsequent update jobs or if you are doing the backouts in the correct order. One customer managed to run Job 1, Job 2, and Job 3 and then backed out Job 2 followed by Job 3.

You will also find that DBRC=C requests EXCLUSIVE authorization on all databases in the PSB, even if the original job only had READ or READ-ONLY authorization.

APAR Number: PN74495

One other problem with DBRC=C is that it allows incorrect input to batch backout, as the text from the above APAR shows.

"The use of DBRC=C allows Batch Backout to backout a job which has completed normally. Since such a job completed normally, the databases involved could have been accessed and modified by subsequent jobs, either in the same or another IMS subsystem. In the case of the same subsystem, DBRC will fail log verification because the logs are not the "last" logs for the subsystem. In the case of a modification by another subsystem, DBRC will not return any of the logs for another subsystem and the utility may verify the logs correctly when it should not. Both cases are problems where it is the user's responsibility to determine that the correct logs are available and that the databases have not been modified by a subsequent job.

Module DFSBACK0 has been modified to check that a BYPASS LOGVER Utility Control Statement is included in the SYSIN data set when DBRC=C is specified. If BYPASS LOGVER is not included, a new message, DFS3296A DBRC=C REQUIRES BYPASS LOGVER is issued and Batch Backout terminates with a new return code 140. No backouts are done. Module DFSBACM0 has been modified to include the new message.

This APAR fix corrects a deviation from the design specification for DBRC support of the Batch Backout utility. DBRC=C will now work as originally specified and will be so documented. Recognizing that there is a need for additional support in this area, a requirement "DBRC Log Verification support for normally completed jobs", number REQ00053297 has been created. Input to this requirement will be considered for possible future releases of IMS."

Automating Batch Procedures

Here are a couple of problems I came across in a database level sharing environment. The second one also applies to n-way sharing.

Back Out of Multi-Step Batch Jobs

A multi-step job has completed successfully and you want to run multiple backouts with DBRC=C to clear out the effect of the job. However, the job accessed a database (with PROCOPT=GO) that is currently being used online. You will experience problems when running backout in these circumstances, as it tries to get EXCLUSIVE authorization (see above). This problem applies only when trying to backout a normally completed job and does not apply if the job abends.

Authorization Failure in the Middle of a Multi-Step Job

DBRC determines that one or more of the databases are not available (for example, being updated online), and the job abends with an authorization failure. You run backout, the normal operator action for a failed job, which also fails, as it cannot find a X'42' record on the log.

Your procedures must handle the following situations:

- Normal (E)STAE-type abend. Job step updated some databases.

  SUBSYS record exists in RECON and is flagged as ABNORMAL TERM. Run backout for failed step.

- Normal (E)STAE-type abend. Job step updated no databases.

  No SUBSYS record exists in RECON. No backout required.

- Abend due to authorization failure

  No SUBSYS record exists in RECON. No backout required. Backout cannot be run (no records on log).

- Normal non-(E)STAE-type abend (for example, power failure)

  SUBSYS record exists in RECON and is not flagged as ABNORMAL TERM. Either run backout for the failed step or issue CHANGE.SUBSYS ABNORMAL to trigger the (E)STAE processing and see if backout is required. Don’t forget to close the log!

CHANGE.DB UNAUTH

You can use the CHANGE.DB UNAUTH command to remove authorization information from RECON in the rare event where the subsystem record no longer exists in RECON. As they like to say, this should not happen!

Online Backout

In the bad old days, if a transaction could not be backed out, all databases referenced in the PSB were stopped. Nowadays, only those databases that cannot be backed out are stopped (does not apply to CICS).

Dynamic backout can be restarted via the /STA DB command. If the OLDS are no longer available, batch backout is required.

If the database is registered, the /STA DB command checks in RECON to see if batch backout has already been performed. The NOBACKOUT parameter can be used for /STA DB commands on non-registered databases that have already been backed out.

Batch Backout Utility

For online IMS systems, the utility distinguishes between backouts of online backout failures and backouts of incomplete transactions.

Backouts of Online Backout Failures

If control statements are not provided, only databases with online backout failures (dynamic backout, emergency-restart backout, partial backout) are backed out by the utility. The /STA DB command causes dynamic backouts to be retried. The batch backout utility recognizes such occurrences. The input logs must include all logs from the last sync point up to the last successful online backout.

Backouts of Incomplete Transactions

To handle these situations, new control statements must be used:

- COLDSTART

  Designed for running backout after an IMS failure (for example, after an /ERE failure). See later in this chapter for additional information.

- DATABASE

  Designed for situations where there are more than 11 input logs. You should avoid this by checkpointing regularly. Actually, what you should do is shoot any programmer who causes this problem!

Multiple logs must be input in the order of oldest first. Multi-volume DASD logs must be entered on separate DD statements to backout (i.e., one DD statement for each volume).

Issues and Concerns

Batch backout is the major area in DBRC where you are eminently capable of getting it wrong because DBRC, in my experience, does not check everything that it should.

- DBRC is meant to verify that the log is closed prior to backout, but I've met customers who have had problems. One customer reported running backout for an 800-cylinder unclosed log. Backout only backed out the first track (with condition code 0) because the DSCB contained zeroes (the first track is relative track zero).

- DBRC does not verify that you are using the correct log. It only verifies that the log was produced by a job of the same name as the one that abended. You could easily use yesterday's log by mistake.

- DBRC does not store the PSBNAME in RECON.

- DBRC (and IMS) does not verify that a batch job is restarted at the checkpoint from which it was backed out. BMC Software's APPLICATION RESTART CONTROL can cure this problem.

- If you cannot run backout, or if you prefer to use some other means of cleanup (for example, restore and rerun), you have to issue a series of commands to DBRC to clean up the RECONs before you can rerun the job.

- DBRC does not verify that you are backing out jobs in the right order (DBRC=C).

Warning! It is safe to close a closed log. It's safe to rerun BACKOUT. It is probably going to corrupt your databases if you forget either.

Keep all log data set names in RECON unique; i.e., do not reuse a data set name until you have cycled through your image copy GENMAX and issued a DELETE.LOG INACTIVE to clean out old log records. The reason for this is that if you ever run DFSULTR0 to close a log, it will find the first matching log in RECON, which may or may not be the one you are trying to close.

Also, one customer managed to reuse the same generation and VOLSER in a batch job as had been used the week before. The job abended and backout was run—unfortunately, it backed out last week's updates because they used the wrong log!

IMS V4 Backout Support

IMS V4 introduced new support for BACKOUT. Batch backout calls DBRC and provides a timestamp and a database data set name. DBRC determines the complete set of logs required. There is still no GENJCL.BACKOUT.

All this code was unfortunately designed for online backout failures and not for batch. Use the techniques outlined in this chapter or BMC Software's BATCH CONTROL FACILITY.

```
BACKOUT
 SSID=SYS3      #UORS=2

   RECOVERY TOKEN=E2E8E2F3404040400000000300000002
   TIME=1997.274 09:13:27.0 -09:00       PSB=PLVAPZ12
                        INFLT   BMP   COLDEND
    ASSOCIATED DATA BASES=3

                 BACKED   DYN BKOUT
    -DBD-        -OUT -   -FAILURE-
    DHVNTZ02      NO         NO
    DXVNTZ02      NO         NO
    DIVNTZ02      NO         NO

  RECOVERY TOKEN=E2E8E2F3404040400000000400000000
  TIME=1997.274 09:13:28.0 -09:00       PSB=PSBEJK05
                       INFLT   BMP   COLDEND
   ASSOCIATED DATA BASES=2

                BACKED   DYN BKOUT
   -DBD-        -OUT -   -FAILURE-
   DBOHIDK5      NO         NO
   DXVHIDK5      NO         NO
```

Figure 35 Sample BACKOUT Record

The DBRC validation for batch backout in IMS V4 for batch jobs is as follows:

- DBRC will validate that the input log to batch backout is the latest “non-backout” log created by the subsystem when a batch job is being backed out. (Note—the subsystem name for a batch job is the job name used when the log was created). DBRC will also validate that the log has been closed (non-zero STOPTIME).

- Batch backout asks DBRC if the log for the subsystem being backed out is valid. DBRC determines the subsystem name by reading the X'42' record on the input log. Thus, if you mean to backout job X, but give batch backout a log created by batch job Y, DBRC will perform validation for subsystem Y and will ensure that the latest “non-backout” log for subsystem Y is being used.

- A “non-backout” log means a log that was not created by batch backout. Logs created by backout have a BBO field in the PRILOG record.

- The DBRC validation operations can be suppressed by supplying a BYPASS LOGVER control statement to batch backout, for example if you were using non-unique job names. This strikes me as a bad parameter which should be avoided!

Warning! This does not fix all the problems outlined in this chapter!

Some Examples of Despair and Misery

Following are some examples of how things can go wrong.

- Job 1 was run with a log and DBRC turned off. Job 2 was run with a log and DBRC turned on. Job 2 abends, but backout is run by mistake with the log from Job 1.

  Backout reads the log, determines that DBRC was not active, and does not look in or update RECON! DBRC still has the SUBSYS record and the authorizations, and hence says backout is still required.

- Step 1 uses a non-registered test database with DBRC active and abends. The SUBSYS record is retained to enforce backout. Step 2 ran backout against the production databases (with dynamic allocation) that were online! Backout does not request authorization since SUBSYS is already there, and does not check data set names for the databases.

The morals here are to FORCE DBRC on everywhere and FORCE registration of your databases. If the job(step) ran with DBRC, run backout with DBRC. If the job(step) ran without DBRC, then run backout without DBRC.

- Job runs with dual logs. Primary log fills up. Job continues to log to secondary log and then abends. Backout was then run with the primary log as input—DBRC does not object, because it is unaware that the primary log is bad.

Note: BMC Software's BATCH CONTROL FACILITY (as you've probably guessed by now) solves this problem and all the other common errors raised in this chapter.

IMS Scheduling

Prior to IMS V2, if a database was not available, all PSBs that referenced that database were not scheduled. Some applications have databases defined in their PSBs that are not used in every scheduling, so V2 changed the scheduling rules.

Scheduling Transactions in the Old Days

With IMS V1.3, transactions were not scheduled if:

- the PSB contains a PCB for a stopped, locked, or /DBRed database
- the PSB wants to access a database that has failed authorization
- the PSB has update intent against a database marked as READ (for example, via /DBD)

Scheduling Transactions Nowadays

As from IMS V2, these checks are not made at scheduling time, but during the application program. These changes do not apply to Fast Path. Fast Path programs schedule against a stopped area, but not against a stopped DEDB or main storage database (MSDB).

The flow now is as follows:

1. Always attempt to schedule transactions (except Fast Path stopped databases).

2. When a DL/1 call requires an unavailable database:

   - NA is returned on an INIT DBQUERY call—see below.

   - If the program has issued the new INIT STATUS GROUPA call, then return a status code—BA or BB.

   - If the database is unavailable because dynamic allocation fails, return an AI status code.

   - Otherwise, pseudo-abend (U3303) and place the input message on suspend queue. Increment abend counter by one.

3. If the program executes successfully, decrement counter by 2 (the counter never goes negative).

4. If counter reaches 10, USTOP the transaction (Unavailable STOP). Reset counter to zero.

5. /STA TRAN or /STA DB will restart scheduling.

   Note: There is no single command that will display those transactions (and associated locked response-mode terminals) that are not being processed because a database is not available.

The INIT Call

The INIT call has two versions.

- It is used with STATUS GROUPA to show that the application can handle BA and BB status codes.

  This version was designed for use by application programmers.

- It is used with DBQUERY to check the availability of the databases.

  This is performed implicitly by IMS before the application is given control. This call can also be issued by an application, but it is not required since it is done by IMS up front. It sets a status code of

  — NA—not available
  — NU—not updatable
  — ' '—available and updatable

  in each PCB and fills the SEGNAME field with the database organization.

SERIAL

You can code the SERIAL parameter on the TRANSACT macro in the IMSGEN. Then IMS will work in the same way as it used to in V1.3, (i.e., abend the transaction once and not schedule any more). However, this also means that you cannot run the transaction in more than one region; SERIAL means no parallel scheduling.

/ERE Failure

If emergency restart fails, try it again. This is why IBM has given you a selection of parameters to try out—COLDCOMM, COLDBASE, as shown below. This chapter assumes that IMS will not restart and you are going to have to perform an unplanned cold start.

Unplanned Cold Start

The following explains how to perform an unplanned cold start.:

Step 1 Close the OLDS from the previous session if it is not already closed. DBRC will have a stop time of zeroes for the last OLDS if it is not closed.

Use GENJCL.CLOSE for this. GENJCL.CLOSE uses the LOGCLJCL skeletal JCL (DFSULTR0 with CLS from WADS). Make sure that this JCL reflects your WADS configuration.

This close job generates a list of all the PSBs active at the time of failure. It also performs the cleanup of the unfinished “59” chain for Fast Path. More details below.

You may experience problems with the backout list.

- DBNAME list may be incomplete
- Backout may be required
- DFS3272I '47' Log record not found

In these cases, rerun DFSULTR0 using the closed OLDS and previous OLDS as input and specify PSBLIST=YES on the EXEC statement. If the close fails due to lack of space on the OLDS, run DFSULTR0 using DUP with CLOSE.

Step 2 Once DFSULTR0 runs successfully, RECON is updated—OLDS is closed and marked as ARC NEEDED.

Step 3 Perform batch backout for the DL/I databases and PSBs listed.

Specify the COLDSTART parameter on the backout. See “Backouts of Incomplete Transactions” on page 92 for additional information.

Step 4 Cold start IMS (message queues lost).

As from IMS V3, you use the command: /ERE COLDSYS. Alternatively, you can try /ERE COLDBASE to emergency restart data communications (DC) and cold start the database (the databases are protected as DBRC turns on BACKOUT NEEDED flags and counters), or /ERE COLDCOMM to emergency restart the database and cold start data communications.

The COLDCOMM, COLDBASE, and COLDSYS options do not affect the handling of indoubt UORs at restart time. The handling is the same as it is for an /ERE restart without specifying any of these parameters. With or without these parameters, IMS will read its log records to determine which UORs might be indoubt to DB2* and inform DB2 whether they should be committed or backed out.

In other words, these options are not equivalent to an IMS cold start in the handling of the DB2 interface at restart time.

BMC Software’s Q:MANAGER IMS EP™ helps you resolve message handling problems through advanced features for dequeuing, unloading, and requeuing messages as well as displaying messages, message queue statistics, and message queue contents. It’s a handy product to have when you are faced with cold starts and emergency restarts.

Fast Path Considerations

In Fast Path, the chain of log records associated with a particular transaction may span OLDS, i.e., half the chain on one OLDS and half on the next. Complete chains indicate completed transactions. An incomplete chain indicates that the transaction did not reach sync point and may be discarded. As from IMS V3, the recovery utility ignores incomplete chains, so simply close the OLDS with DFSULTR0 and then continue as described in the following paragraphs.

Having closed the log, you now have to reset the Fast Path databases. There is no such thing as backout for Fast Path because Fast Path does not log *before* images.

The steps for recovering Fast Path areas are as follows:

Step 1 In Share Control, issue CHANGE.SUBSYS ABNORMAL. This allows DEDB forward recovery to be run.

Step 2 Issue CHANGE.DBDS RECOV for all authorized areas.

Step 3 Perform full recovery of all areas with the RECOVERY NEEDED flag on.

Step 4 Rebuild MSDBINIT from the latest MSDBCPx and the last SLDS.

Step 5 Cold start /ERE COLDSYS.

Step 6 Run the online DEDB create utility to re-establish MADS.

An alternative to this long-winded process of recovering all the Fast Path areas is provided by the TRIMAR FAST RECOVERY UTILITY. See Chapter 13, “Fast Path Utilities” for additional information.

Chapter 9 Image Copy

The basic principle of protecting IMS databases against physical errors is that of taking regular copies plus logging all the changes that occur. (Actually that should be the principle of any DBMS.) This chapter looks at how DBRC records copies and how it selects the correct logs for recovery.

Note: Concurrent image copy (covered later) is the default on the GENJCL command as from IMS V4.

Image Copy Options

There are many different ways of copying IMS databases nowadays—software and hardware. DBRC recognizes many of these. Some of them, however, have no DBRC interface at all, which means that I personally wouldn't contemplate using them in a production environment. For me, image copy should be a fully automatic procedure, submitted by a job scheduler and recorded in RECON, with no manual intervention. It should produce standard copies, and I don't want to issue any strange complicated commands when I come to run a recovery as I am probably in a state of panic anyway.

Offline (Batch) Image Copy

The standard utility recommended for offline image copy is the IMS Image Copy utility or BMC Software's IMAGE COPY PLUS. See Chapter 12, "BMC Software and DBRC" for additional information. The use of non-standard utilities, which normally require manual intervention in DBRC or additional coding, is not recommended.

In this case, the database is taken offline from all other processing while the copy is running so the database is not being updated. This means that the copy is a true snapshot of the contents of the database. Offline image copy requests READ authorization from DBRC. DL/1 utilities only partake in database level sharing and not in n-way sharing. So DBRC only allows concurrent read or read-only here.

Before running an offline image copy, ensure the database is not being updated. DBRC Share Control allows authorization only if the database is not being updated. To achieve this, allow any batch jobs to terminate. For online systems, either:

- Shut down the online system(s).

  OR

- Issue the /DBR command or CICS equivalent.

  OR

- Issue the /DBD command or CICS equivalent.

These commands flush all the buffers associated with the database (i.e., all updates are written to the log), close and deallocate the database, and write a DEALLOC timestamp in RECON. If you did not use the NOFEOV parameter, the commands close and switch the log. In an n-way sharing environment, you use the GLOBAL parameter on these commands to take the database(s) off all sharing subsystems. Or use BMC Software's RECOVERY MANAGER for IMS to automate and receive positive response on the /DBR, /DBD, and /STA commands.

The /DBR command takes the database completely offline. Transactions cannot be scheduled against the database. The /DBD command puts the database in INQUIRY-ONLY mode on the online system and only allows read or read-only (PROCOPT = G or GO) transactions to be scheduled.

DBRC records this in an image copy record with a type of BATCH. DBRC uses the RUN timestamp to select logs for recovery (see log selection section below). RUN time is start of job, and STOP time is stored as zeroes.

```
IMAGE
 RUN     = 1997.274 08:49:59.2 -09:00     *  RECORD COUNT =31
 STOP    = 0000.000 00:00:00.0 +00:00        BATCH      USID=0000000004

IC1
 DSN=IMSVS.DBVHDJ05.CJVHDG1E.IC.IC094956          FILE SEQ=0001
 UNIT=SYSDA                          VOLS DEF=0001 VOLS USED=0001
                                     VOLSER=222222
```

Figure 36 Sample Batch Image Copy Record

Concurrent Image Copy (CIC)

The concurrent image copy (CIC) feature is available for Fast Path DEDB (and DL/1 as from V4) using parameter CIC. It is also available with BMC Software’s IMAGE COPY PLUS, where it is called online image copy. See Chapter 12, “BMC Software and DBRC” for additional information.

CIC runs as a batch job reading the database (without integrity—READ ONLY authorization requested) while the database is being updated. The copy is *fuzzy* and cannot be used by itself. Recovery requires this copy plus the logs that recorded the changes during the copy (log selection is actually more complex than this and is covered later in this chapter). DBRC records this as a CONCURrent image copy with the RUN and STOP times of the job(step).

Note: You cannot copy KSDS this way—you can only do OSAM and ESDS. BMC Software's IMAGE COPY PLUS Snapshot Copy feature (see below) can handle KSDS as well.

You can't use concurrent image copy on a NONRECOV database.

Fast Path has been able to do concurrent copy for years, but you were advised to take the database offline and start it again before you started the copy. This is no longer necessary in V4. IMAGE COPY PLUS fully supports Fast Path DEDBs.

```
IMAGE
 RUN     = 1997.274 08:52:02.7 -09:00     *  RECORD COUNT =31
 STOP    = 1997.274 08:52:19.4 -09:00        CONCUR     USID=0000000005

IC1
 DSN=IMSVS.DBVHDJ05.CJVHDG1E.IC.IC095155          FILE SEQ=0001
 UNIT=SYSDA                          VOLS DEF=0001 VOLS USED=0001
                                     VOLSER=222222
```

Figure 37 Sample Concurrent Image Copy Record

IMS Online Image Copy

The IMS online image copy utility runs as a BMP and copies the database while it is being updated in the same online IMS system. No other update is allowed. This means it is inherently slow and contends with your online transactions for buffers and resources, etc. This utility performs a special sign-on to DBRC and runs as a subsystem with an authorization level of online image copy (Encoded State = 5). Again, the copy is fuzzy and logs are required for recovery.

Online image copy runs as a BMP with a special subsystem ID. This is the only case where a dependent region in an IMS online system interfaces directly with DBRC. Special authorization logic ensures that only the host online subsystem (i.e., the one where the BMP is running) can update the database. If some other subsystem is currently authorized with update (UP) or exclusive (EX) access, the online image copy authorization request will fail.

This utility is not recommended as it is slow, a pain to restart, requires a PSB, has restrictions on how many databases it can handle, and slows down your online system. It was interesting 15 years ago, but things have moved on.

DBRC records this as an online image copy. The image copy time is recorded as the last system checkpoint prior to the start of the online image copy.

```
IMAGE
 RUN     = 1997.274 08:52:02.7 -09:00     *  RECORD COUNT =31
 STOP    = 1997.274 08:52:19.4 -09:00        ONLINE USID=0000000005

IC1
 DSN=IMSVS.DBVHDJ05.CJVHDG1E.IC.IC095155          FILE SEQ=0001
 UNIT=SYSDA                          VOLS DEF=0001 VOLS USED=0001
                                     VOLSER=222222
```

Figure 38 Sample Online Image Copy Record

Incremental Image Copy

A further possibility is to eliminate any down time on the database by taking a previous image copy and applying the subsequent changes to it to create a new image copy. There is no access to the database itself. This function is available with BMC Software's IMAGE COPY PLUS and CHANGE ACCUMULATION PLUS products, where it is called incremental image copy (this has nothing to do with DB2 incremental copies—this is a complete copy, created in the background).

If the change accumulation is run up to the end of a log data set (i.e., there were no OPEN logs), it is complete and the incremental image copy is a valid full copy.

If the change accumulation is run when there is one or more OPEN logs (e.g., the middle of an online IMS or CICS session), the incremental image copy is incomplete, so the image copy is fuzzy and notified with RUN and STOP times (the exact rules are given in the *IMAGE COPY PLUS User Guide*).

Non-Standard Image Copy

Various pieces of software exist to do this but they do not take proper IMS image copies, i.e., the copy is *not* in IMS image copy format, is not registered in RECON, and cannot be used as input to the recovery utility. In fact, recovery has to be run as two steps:

1. Restore the copy using the appropriate utility.

2. Run the IMS Recovery utility to apply the logs/change accumulations on top of the restored database data set.

   Note: BMC Software's RECOVERY PLUS can do this step.

This means recovery will be slower and prone to error. Other disadvantages:

- The copy cannot be pointer checked/analyzed while it is being taken.

- The user must guarantee that no updates are happening to the databases on the pack(s) being copied.

- If you are using DBRC, then you must issue NOTIFY the copy to DBRC as a user image copy (UIC). This is a horrible manual interface, and the recovery job becomes even more complex:

  — Restore the copy.

— Tell DBRC you have done this via a command, which requires you to look up the accurate twelve-digit timestamp of when you took the copy.

— Run a recovery applying the logs/change accumulations on top of the restored database data set.

It is, of course, designed to recover packs—not the data sets that you are interested in.

Use of utilities that do not automatically update RECON are not recommended. If you do use one of these, notify DBRC of the copy you have taken. Do this using the following command:

```
NOTIFY.UIC DBD(...) DDN(...) CURRENT|RUNTIME(....) UDATA(....)
```

Note: The timestamp used to identify when the copy was taken is required to restore the copy. See Chapter 11, "Recovery" for details.

You can only notify DBRC of user image copies if the DBDS has been registered in RECON as NOREUSE.

You cannot register *fuzzy* user image copies—UICs must be consistent.

IBM 3990 DFSMS/MVS Concurrent Copy

Prerequisites to IBM 3990 DFSMS/MVS Concurrent Copy are the IBM 3990-3 with Extended Platform, DFSMS/MVS 1.1, and MVS/ESA SP Version 4. It takes a point-in-time copy of a data set—using the same concept as BMC Software's Snapshot Copy (for details, see below), i.e. it looks at before images, but it uses the cache in the hardware.

The copy is *not* in IMS image copy format, and hence has the same problems as above.

- The copy cannot be pointer checked/analyzed while it is being taken.
- The copy will also be "fuzzy." This means that the user is responsible for selecting the correct logs for input to the recovery, and all of this has to be done outside DBRC control because you cannot register "fuzzy" user image copies.
- Or, the user has to /DBR the database before the copy starts, wait for the copy started message and then /STA the database again (recommended method by IBM). This is all manual.

- If you are using DBRC, then you must issue /DBD or /DBR and NOTIFY the copy to DBRC as a user image copy (UIC)—same problems as above. This is cured in Version 6.

IBM IMS Image Copy 2—(available with IMS V6)

Starting with IMS Version 6, IMS recognizes this IBM 3990 DFSMS/MVS concurrent copy as a new type of copy (T0 copy) and records it in DBRC (SMSCIC or SMSNOCIC depending on whether it is fuzzy or consistent), but it still has a bunch of problems:

- not a true image copy
- requires manual DBD/DBR and START commands as above
- only does one image copy per step
- limited by what the hardware cache can handle
- cannot be pointer checked/analyzed at the same time
- no support for multi-volume databases
- database must be registered to use this option

IBM RAMAC SnapShot

Prerequisites to IBM RAMAC SnapShot are:

- IBM RAMAC Virtual Array Model 2, IBM RAMAC Virtual Array or StorageTek Iceberg disk arrays

- IXFP version 2.1

- SnapShot version 1.1

- RAMAC Virtual Array enabled for SnapShot

- All IMS database data sets and FP DEDB areas must be on the same RAMAC Virtual Array subsystem

- There must be sufficient functional volume space to hold the database data sets, FP DEDB areas and copies of these IMS components

SnapShot (which is not the same as BMC Software’s Snapshot Copy, as is described on page 109) is a virtual data duplication product. This virtual data duplication capability (Snap) is exclusive to the virtual storage architecture. In this architecture, virtual volumes are represented as a set of pointers in tables. These pointers are stored in non-volatile storage (NVS) within the device.

A SnapShot is simply the creation of a new virtualization of the volume being snapped by copying the pointers.

It does not take proper IMS image copies, i.e., the copy is not in IMS image copy format, is not registered in RECON and cannot be used as input to the recovery utility. In fact, recovery has to be run as two steps:

1. Restore the copy using the appropriate utility.
2. Run the IMS recovery utility to apply the logs/change accumulations on top of the restored database data set.

   Note: BMC Software's RECOVERY PLUS can do this step.

This means recovery will be slower and prone to error.

Problems:

- copies at the volume level for IMS and DB2 databases, not at the data set level
- data set level copy not supported for IMS and DB2 data sets
- is a virtual copy of the volume, not a true image copy
- requires manual DBD/DBR and START commands as above
- cannot be pointer checked/analyzed at the same time
- copy on same device as database—what happens if you lose device?
- manual NOTIFY.UIC to DBRC—in fact, the operator is told to record the 12-digit timestamps as he or she goes along because they will be needed for the DBRC commands later!
- non-standard recovery (snap back procedure), manual NOTIFY.RECOV to DBRC

EMC Symmetrix (Timefinder)

Symmetrix Remote Data Facility (SRDF) maintains a real time copy of the data at the logical volume level in two to five Symmetrix 5xxx systems located in physically separate sites.

This is basically a data duplication and mirroring capability. IMS concerns with this technique are similar to those for other DASD volume level backups.

The problems with this are the same as the RAMAC SnapShot above. The hardware in both cases is very clever and very fast at making a copy within the same device, but it requires manual operator intervention, has no DBRC interface, and basically does not understand databases at all. It requires someone (like BMC Software) who understands databases to automate it all. See also “Image Copy” on page 165 for further problems that can occur.

Snapshot Copy

Snapshot Copy is a function of BMC Software's IMAGE COPY PLUS utility. The Snapshot Copy feature lets you interface with BMC Software's EXTENDED BUFFER MANAGER (XBM™) for IMS or BMC Software's SNAPSHOT UPGRADE FEATURE (SUF™) component to create a Snapshot Copy (true batch image copy) of registered databases. The Snapshot Copy function copies a single database or a group of databases at the same consistency point while the database remains online for updates.

This means that you are producing a consistent (in DBRC terms BATCH) copy of the database, while it is being updated. Furthermore, this means that you can run BMC Software's POINTER CHECKER PLUS® or FAST PATH ANALYZER/EP™ at the same time to carry out consistent checking of your database. Today, if you run concurrent copy or any other type of “fuzzy” copy, then POINTER CHECKER PLUS or FAST PATH ANALYZER/EP will probably show that there are errors in the database (because the database is changing during the copy and hence the check is not being performed on physically consistent data). With Snapshot Copy, this problem goes away, if POINTER CHECKER PLUS or FAST PATH ANALYZER/EP says you have a problem, then you really do have a problem. Also, as stated above, you can copy KSDSs safely with this method.

BMC Software’s SECONDARY INDEX UTILITY/EP

BMC Software SECONDARY INDEX UTILITY/EP has the ability to recreate both secondary indexes and hierarchical indexed data access method (HIDAM) primary indexes. Many customers have now stopped image copying their indexes and use SECONDARY INDEX UTILITY/EP in the event of an error. See Chapter 12, “BMC Software and DBRC” for additional information.

MADS and Online Create

Fast Path DEDBs can be defined to have multiple copies of one or more areas of a DEDB. If you use this option, you can also use the online create utility to create a new copy of an area data set in-flight while the DEDB is being updated.

You are still recommended to take image copies with MADS for two reasons:

1. Offsite backup

   Batch (or BMC Software Snapshot) image copies are the only way to take a consistent copy offsite.

2. ALLOC/LOGALL cleanup

   If you don't take image copies, DBRC cannot perform its cleanup. As from V3, you can define DL/1 databases as NONRECOV to avoid this problem. See "RECOVABL/NONRECOV" on page 42 for additional information. Unfortunately, you cannot currently define DEDBs as NONRECOV.

DBRC Log Selection

This section considers which logs DBRC selects, depending on the type of image copy you are using. For the moment, I will just look at full forward recovery. Timestamp recovery is covered in "Timestamp Recovery" on page 124.

Warning! Don't try and select logs yourself—let DBRC do it for you.

Batch Image Copy

In this case, change accumulation selects all logs which start after the image copy is finished.

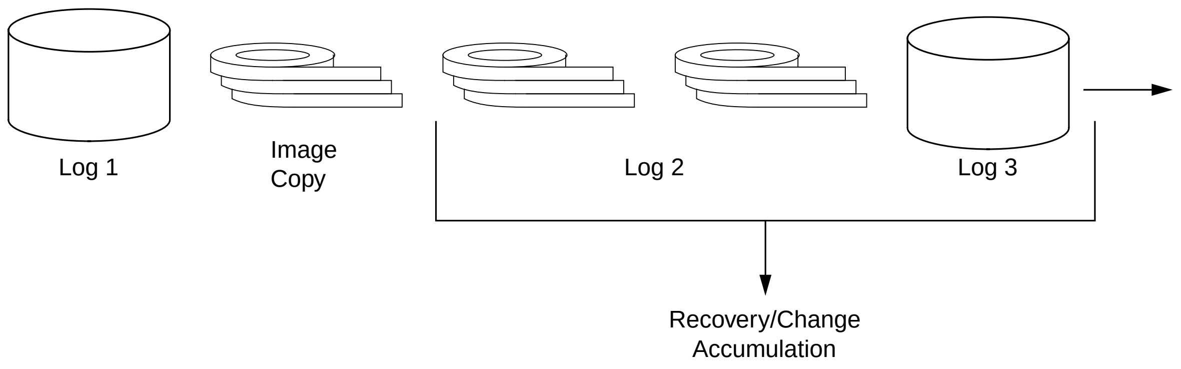


Figure 39 Change Accumulation/Recovery with Normal Operations

In a block-level sharing environment, you will have overlapping logs with updates.

1. DBRC realizes this and forces change accumulation prior to recovery.

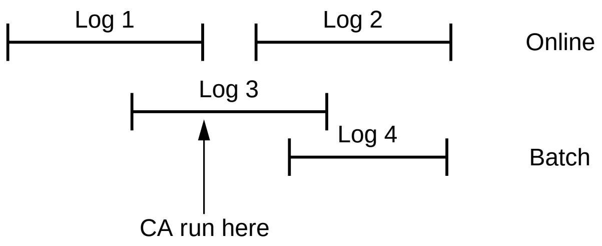


Figure 40 Change Accumulation with Block-Level Sharing

2. DBRC change accumulates what it can and produces spill records in the CA output data set for those records it cannot change accumulate because the overlapping log is still in use (LOG3 above). These will get change accumulated as soon as you run a CA including the relevant logs. DBRC flags CAs without spill records as complete.

   The good news is that you can run change accumulation whenever you want, and DBRC will force you to run it prior to recovery, if required. See Appendix D, “IMS Change Accumulation Utility” for additional information. Alternatively, if you have BMC Software’s RECOVERY PLUS product, it will accumulate any necessary logs as part of the recovery.

This means that in the diagram, change accumulation can be run at the end of any of the log volumes shown. In the event of recovery, the database must be taken offline (e.g., via /DBR for online IMS) from all subsystems and all the logs since the last image copy/change accumulation must be included in the recovery.

Fuzzy Copy

In all the cases where you take a fuzzy copy, this is not a valid recovery point—the database is changing while you are copying it. Therefore, any recovery is going to involve not only the image copy, but also the logs. You are strongly recommended to use DBRC and let it find the correct logs for you. (Or use the Check Assets function in BMC Software's RECOVERY MANAGER for IMS to get a list of inputs required by DBRC for recovery.) I was going to document all the rules for log selection here, but they are tedious, boring, complex and I am not sure I would get them right—so use DBRC instead.

Chapter 10 Change Accumulation

Change accumulation does not read databases, so it is not a subsystem. Because of this, it does not sign on to DBRC or request authorization. However, DBRC checks the input and output data sets and records the execution.

The advantage of change accumulation is that it sorts and merges all the changes from the logs, so if a segment has been updated twenty times, only the last change is required for recovery and that is the one stored on the change accumulation data set, which makes recovery faster. The disadvantage is the extra workload.

This chapter looks at how change accumulation works with DBRC and how to reduce this workload.

Change Accumulation Groups

One of the neat features that comes with DBRC is the ability to define multiple change accumulation groups (CAGRPs). Before talking about CAGRPs, I would first like to consider why people (should) run change accumulation at all.

- To reduce image copy frequency

  If you use change accumulation, you can leave a longer period between image copies without impacting recovery times. This can lead to major reductions in the number of cartridges required for backup, and help to relieve the pressure on the batch window.

- To reduce recovery times

  The recovery utility will run faster if you use change accumulation as input. (The change accumulation utility is basically a sort process which throws away all of the database updates except the latest ones and then sorts the updates in the same order as the image copy). In this case, the recovery utility merges the image copy and the change accumulation file and then applies any logs in a separate second stage.

  If you use the IMS utility and input one or more logs, then each individual database change record on the input log(s) will be handled via DL/1—a very slow process. The IMS utility is also designed to only recover one database data set per run, so you will also end up reading the logs over and over again.

  If you use BMC Software's RECOVERY PLUS, then you will only read the logs once even when recovering multiple databases. However, the BMC Software utility will also run faster with change accumulation input because it now has much less input data to read and does not have to do any sorting.

- To prove the logs are readable

  It is embarrassing (and potentially extremely time-consuming) to find out your logs are unreadable when you come to run a recovery.

- To reduce the number of logs/amount of first level recovery data

  Many customers use the technique of logging to disk in batch and/or archiving to an RLDS on disk for online. They then run change accumulation and let HSM migrate the input log(s) to a cheaper medium. Hence, the “shelf life” of a disk log/RLDS is relatively short and you free up expensive DASD for the next job. Use of IDRC tape drives for the migration gives the added benefit of compressing the logs. Change accumulation can also use compression for its output data set as from IMS V3.1 or when using BMC Software's CHANGE ACCUMULATION PLUS.

- Because you are running n-way sharing or using /DBR NOFEOV

  If you are running sharing (or take databases offline with /DBR NOFEOV and then run batch work), you will have overlapping logs with updates for the same databases. The log records in recovery have to be in timestamp order, so you have to sort them prior to recovery. To do this you either use change accumulation/CHANGE ACCUMULATION PLUS or BMC Software's RECOVERY PLUS (which also sorts log records).

Note: Even if you are not using sharing, you are recommended to run a log sort process prior to recovery—I have come across scenarios where not using DBRC and not sorting logs leads to database corruption.

Why Should I Use CAGRPs?

If you run change accumulation without DBRC, then you code up control statements to say which database data sets should be accumulated together in one execution. Hence, you can accumulate all of them or you can do a subset at a time.

With DBRC, you have to define one or more CAGRPs. A CAGRP is simply a list of the database data sets you wish to accumulate together in one execution of change accumulation. You can have one group with all the database data sets in it or you can have multiple groups.

The reasons for having multiple groups are:

- Faster recovery

  Having groups means that each of the change accumulation files will be smaller and, hence, the recovery utility has less data to read.

- Parallel recoveries

  If you have multiple groups, then you can run parallel recovery jobs—one for each group.

- Faster change accumulations

  Each change accumulation execution will be quicker as there is less data to sort. This also has major benefits in terms of SORTWRK space—lots of small sorts run much faster and use much less space than one big sort.

Group Rules (OK?)

A change accumulation group is a collection of database data sets processed in a given execution of change accumulation. Change accumulating everything using DB0*ALL is not possible with DBRC. Change accumulation is always run on a group basis (the group can, of course, contain all your database data sets) when using DBRC.

A single change accumulation job cannot handle a mixture of registered and unregistered databases. A given database data set can only belong to a single change accumulation group.

A change accumulation group in IMS V2 can contain up to 1024 members (= database data sets). In V6, this was increased to 2000. The ADD and DELETE operands on the CHANGE.CAGRP command allow you to add new members to an existing change accumulation group or delete members from a change accumulation group without having to delete the entire change accumulation group. You are recommended to copy any member you move or add—if you don't, then the next time you run change accumulation for that group, it will pull all the logs in again to give itself a valid change accumulation output data set.

Do not change the members of a CAGRP while a change accumulation for this group is running. It corrupts the CA record in RECON!

The main problem with the standard change accumulation utility is the following:

Assume that there are 600 database data sets accessed by the online system. You define six change accumulation groups with 100 in each because that is the way your applications split down. Change accumulation is basically a sort, and it is more efficient to run six small sorts than one big one. This means that to change accumulate all of the database data sets, you must run change accumulation six times—once for each group. Unfortunately, DBRC is not clever enough to allow these to run in parallel, and each change accumulation needs a pass of the logs. If a single change accumulation group with 600 members is defined, the logs are only processed once, but the sort is not nearly as efficient and uses all your sort workspace. Hence, many customers drop the idea of running change accumulation at all, or only in the event of recovery, which is unfortunate as regular change accumulation can decrease the need for image copy and speed up recoveries.

The solution to this problem is BMC Software's CHANGE ACCUMULATION PLUS, which handles multiple groups in parallel with one pass of the logs. In a recent test, CHANGE ACCUMULATION PLUS handled 96 input logs with 10 CAGRPs in 1.25 hours. The standard utility required 20 hours for each group.

Change Accumulation Rules

When using the standard utility, use GENJCL to generate the job because every time change accumulation is run, the JCL is different. With BMC Software products, there is no need to use GENJCL—see “GENJCL and BMC Software” on page 12.

Logs are not included as input to a change accumulation group if they were created prior to the latest image copy. DB0 control statements are needed for each member in the change accumulation group. These are created for you by DBRC and include purge times. The purge time is equal to the latest image copy time. Any log records older than the purge time are thrown away, because these changes are already on the image copy; this way change accumulation is more efficient.

Warning! Do *not* touch these purge times or try to set them yourself. If you put the wrong purge time in, you may throw away records required for recovery and corrupt your database.

When change accumulation runs successfully, it creates a change accumulation record. This record contains information such as the change accumulation data set name and VOLSER(s). It also records which log volumes participated in this particular change accumulation.

You should modify the skeletal JCL that is provided with DBRC. The skeletal JCL provided requires a tape drive for every tape volume that change accumulation reads. Use the UNIT=AFF technique to reduce the number of tape drives required. For example:

```
//DUMMYDD  DD DSN=DUMMY.TAPE.%time,UNIT=(TAPE,,DEFER),DISP=NEW
//DFSULOG  DD DSN=UTILITY.%logdsn,UNIT=AFF=DUMMYDD,....
```

Figure 41 UNIT=AFF in CAJCL

If you are using CHANGE ACCUMULATION PLUS, then you are recommended to use dynamic allocation in the JCL and all of this is handled for you automatically.

If you are using Fast Path, you need to update the ID control statement in the skeletal JCL. See the *Utilities Reference* manual for details.

DBRC does not record any data set(s) created by the DB1 control statement(s). It is recommended you use RLDSs instead. See Chapter 7, “Bad Data Sets” for instructions on what to do if there is a bad (unreadable) RLDS or change accumulation data set.

Change Accumulation Strategies

There are various change accumulation options open to you. The one you choose depends on your environment, volatility of databases, criticality of recovery, etc.

- Regular change accumulation

  This change accumulation option provides rapid recovery, but requires a regular workload. Keep groups to a minimum, unless you have CHANGE ACCUMULATION PLUS.

- Run change accumulation only in the event of recovery

  The overall recovery time is longer than the previous case, because you have to run change accumulation first, but this option requires no regular change accumulation workload.

- Temp CAGRP

  Combine this with the previous option. Only define a group when required, run change accumulation, run recovery, and delete the group.

- Do not run change accumulation

  Use logs in recovery. Recovery will be much longer because logs use the DL/1 buffer handler (applying each change individually) whereas change accumulation uses native access methods.

Also consider BMC Software’s RECOVERY PLUS product, which accepts image copy, change accumulation and log input, and is much faster than the IMS recovery utility. See Chapter 12, “BMC Software and DBRC” for more information about RECOVERY PLUS.

Common Problems

One problem with DBRC is that if the situation changes between generating the JCL and executing the JCL, DBRC may object to your job and abend it (RC=8). An example of this is if you generate JCL for change accumulation and then before the job starts to execute, the databases are image copied. DBRC now decides there is nothing to accumulate and stops the job. The moral here is to run your change accumulations and then trigger off your image copies. Again, this can be avoided if you use CHANGE ACCUMULATION PLUS without GENJCL, because it generates the JCL internally and executes it all in one go.

One parameter that causes confusion is the CATIME parameter. You can use this to change accumulate up to a certain time. You cannot use this parameter to go beyond an image copy. You can only use it to change accumulate a subset of logs that have not been accumulated yet.

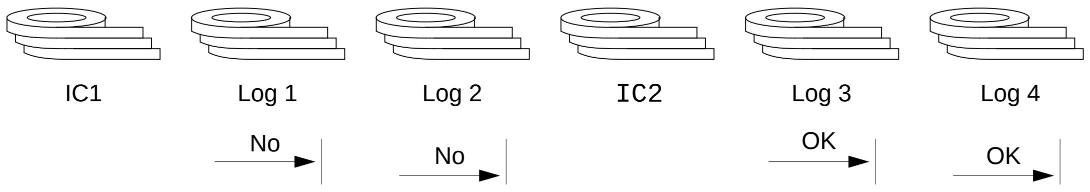


Figure 42 CATIME Flow

For example, you image copy a database (IC1), then you update it on LOG1 and LOG2. Now you image copy it again (IC2) and update it on LOG3 and LOG4. If you tried to do a GENJCL.CA CATIME to the end of LOG1, it would say there is nothing to accumulate because you have image copied the database.

You could use CATIME to the end of LOG3 or LOG4. The only way you can change accumulate to the end of LOG1 is to mark IC2 as INVALID. Now DBRC says none of the logs has been accumulated, and the database has not been copied, so it is valid to run a change accumulation.

See also “DBRC Log Selection” on page 110 and Appendix D, “IMS Change Accumulation Utility” for rules of log selection and details of spill records.

IMS V3 Changes

The recovery utility was rewritten in IMS V3 for Fast Path DEDBs. Prior to V3, the recovery utility would include all DEDB change records it found on the input logs. With V3, the recovery utility ignores any incomplete chains.

The change accumulation utility has also been changed. A CA created under V3 may contain a new type of record, representing uncommitted data (incomplete chains). DBRC lists the CA record as containing IN-DOUBTs. Subsequent change accumulation or recovery will find the rest of the chain on the next log.

This has implications on several areas covered later, namely /ERE failure, disaster recovery, and shadow databases.

Chapter 11 Recovery

The major fault of the IMS recovery utility is that it only recovers one DBDS/area per job step. This means that you have to read the logs once for each DBDS/area. Change accumulation, as discussed in the previous chapter, can help in this process. However, as the recovery utility requests EXCLUSIVE authorization and DBRC only allows one utility with EXCLUSIVE at a time to access a database (full function) you cannot recover multiple data sets or partitions within the same database at the same time with the standard IMS utility. Multiple areas can be recovered in parallel as DBRC authorization is at the area level for DEDBs.

This chapter shows how recovery works and how to achieve parallelism.

Recovery Basics

Three types of recovery are supported by DBRC: full recovery, timestamp recovery, and track recovery.

- Full recovery is normally used for a media failure. Full recovery allows you to recover a database to its original state at the time of failure.
- Timestamp recovery allows you to recover a database to its state at a specified time.
- Track recovery (VSAM only) is used to recover a track in error. Track recovery will not be discussed in this document, because I've only ever found one customer who had tried it (and it didn't work!).

You should always use GENJCL to create the JCL for a full recovery or timestamp recovery (unless you have BMC Software's RECOVERY PLUS or RECOVERY MANAGER for IMS—see "GENJCL and BMC Software" on page 12). The JCL will be different every time you execute a recovery.

You are allowed to put job steps on the front and the back of the particular step you wish to generate. For instance, as the first step of a recovery job you could generate an IDCAMS step to delete/define the relevant DBDS.

```
%DELETE  (%DBDSAM NE 'VSAM')
//STEP0       EXEC        PGM=IDCAMS
//SYSPRINT    DD   SYSOUT=*
//SYSIN       DD   DSN=ARM.IDCAMS.PDS(%DBDDN)
%ENDDEL
```

Figure 43 Sample JCL to Generate an IDCAMS Step to Define a DBDS

Full Recovery

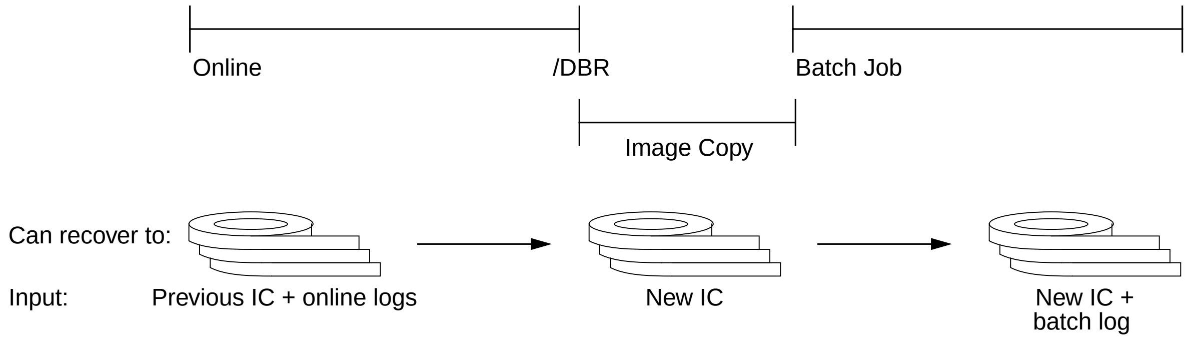


Figure 44 Valid Recovery Points

The inputs to a full recovery are the latest image copy, the latest change accumulation (if one exists), plus any required logs. DBRC checks the logs to determine if a merge situation exists (discussed below). The GENJCL, or the utility itself, will fail if a merge is required. In the case of a failure, run change accumulation first.

BMC Software’s RECOVERY PLUS performs not only the recovery, but in the same pass of the inputs (image copy, change accumulation, and logs) performs any necessary merge, performs an image copy, checks the database (using BMC Software’s POINTER CHECKER PLUS® or FAST PATH ANALYZER/EP™), and rebuilds the primary and secondary indexes (using BMC Software’s SECONDARY INDEX UTILITY/EP). It does this for multiple DBDSs/areas in parallel with one pass of the input data sets, and can recover multiple data sets or partitions in the same database in parallel. It also contains a facility to execute the relevant member(s) of an IDCAMS PDS to delete and define the database data sets.

BMC Software's RECOVERY MANAGER for IMS will create and optimize RECOVERY PLUS and IMS recovery utility JCL for an entire group of databases.

Merge Needed

DBRC recognizes when two or more overlapping logs exist, both containing changes for one or more databases. This happens in block-level sharing and in non-sharing situations where you take a database offline with the NOFEOV parameter and then update it in batch. You cannot input the logs directly to recovery because recovery checks timestamps, which are now out of sequence. So change accumulation must be run first to sort the records in the right order. See Chapter 15, "IMS and the TOD Clock" and Appendix D, "IMS Change Accumulation Utility" for details of how it does this. If you try to run GENJCL.RECOV, DBRC will tell you that a merge is required and does not generate the recovery JCL.

As stated above, RECOVERY PLUS does the merge for you, so there is no need to run change accumulation prior to the recovery.

Recovery Authorization

Recovery requests EXCLUSIVE authorization. For DL/1 databases, the first thing that happens is the RECOVERY NEEDED flag and counter in RECON are turned on (if not on already). Fast Path areas can only be recovered if they are flagged as RECOVERY NEEDED. BMC Software's RECOVERY MANAGER for IMS automates this process for Fast Path areas.

Next, DBRC makes sure any subsystems currently authorized for this database are abnormally terminated. If you fail authorization for this reason, issue a CHANGE.SUBSYS ABNORMAL for any failed subsystems and take the database offline from any running subsystems.

When the recovery procedure successfully completes, a recovery record indicating that a recovery occurred is built and the RECOVERY NEEDED flag and counters are reset for DL/I. Fast Path areas must have the RECOVERY NEEDED flag turned off, and ADS must be made AVAIL. BMC Software's RECOVERY MANAGER for IMS automates this process.

```
RECOV
RUN     = 1997.274 08:31:24.9 -09:00     *  RUN USID      = 0000000001
```

Figure 45 Sample Full Recovery Record

Timestamp Recovery

The only valid times to recover a database using the standard IMS recovery utility are:

- after an offline image copy or an online image copy where there is no overlapping database update (i.e., no overlapping ALLOC)

- at the end of a log data set (caused by shutdown or end of job) containing updates for the database being recovered (i.e., there is an ALLOC record in RECON for this log)

  Note: Do not choose a log that is the end of an abended job. You must choose the end of the backout log in this case.

- at the end of an online log volume caused by /DBR or /DBD for the database in question containing updates for the database being recovered (i.e., there is a DEALLOC timestamp in the ALLOC record in RECON)

  You cannot recover a database to the end of a log unless it falls into one of the above categories. Updates can span log volumes. If this occurs, the recovered database is inconsistent. DBRC enforces this rule.

  IMS V5 adds one new option, which is recovery to the middle of a log, where you have performed a /DBR NOFEOV for the database(s) being recovered. This is covered in detail later in this chapter.

  BMC Software's RECOVERY PLUS supports DL/1 and DEDB and allows you to recover to *any point in time.* BMC Software's RECOVERY MANAGER for IMS helps you pick a good timestamp for a point-in-time recovery. See Chapter 12 "BMC and DBRC" for details.

```
RECOV
RUN      = 1997.275 15:57:15.1*  RUN USID      = 0000000004
RECOV TO  1997.274 18:05:03.2   RECOV TO USID = 0000000001
```

Figure 46 Sample Timestamp Recovery Record

The timestamp you specify can be any time at which the database is deallocated (in DBRC's terms, no ALLOC), and DBRC will work backwards until it finds one of the above conditions. You can also combine this with DBDS groups. You can ask DBRC to recover a group of DBDSs and you only need to specify one timestamp. For example, if you image copied all the DBDSs in the group between 18.00 and 18.20, simply specify 18.20 for the whole group.

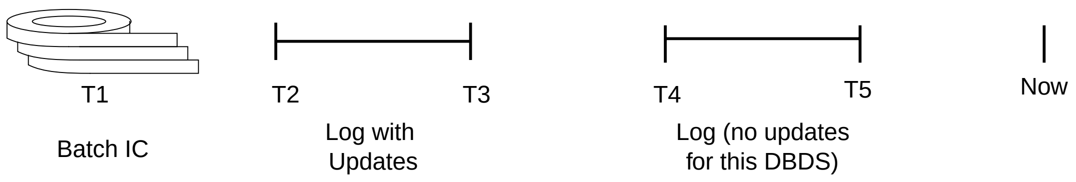


Figure 47 Selecting Timestamps for Recovery

In this diagram you can use

- any time from *T1* to *T2* (would recover from image copy)
- any time from *T3* to *Now* (would recover from image copy and first log)

Note: In n-way sharing or continuous operations environments, it is normally impossible to find one of the above times unless you have RECOVERY MANAGER for IMS. You either have to design your whole operation on the basis that timestamp recovery is not an option, or you have to shut down the system once a day or once a week to give you a recovery point. This is also recommended if you need a point where you can ship data offsite. Alternatively, if you have been kind enough to buy RECOVERY PLUS, then you can recover to *any* point in time even in an n-way sharing environment.

BMC Software's RECOVERY MANAGER for IMS provides a Find Recovery Points function that can be used to find standard recovery points from the RECONs for a group of databases. RECOVERY MANAGER for IMS will consolidate these recovery points and interactively display recovery points in common as well as individual database recovery points. Each recovery point will be categorized as BATCH IC or online/batch subsystem name. RECOVERY MANAGER for IMS will also scan logs to find quiet points of transaction inactivity and display these individual and common recovery points for a RECOVERY PLUS point-in-time recovery.

Using GENJCL.USER to Select Timestamps

You can use the %SELECT logic to fill in the timestamps for you. In my opinion, you should try to avoid ever entering a DBRC command with a 12-digit timestamp. You should automate the interface with DBRC wherever possible, for instance, by using BMC Software's RECOVERY MANAGER for IMS. If, unfortunately, you have not yet installed this masterpiece, then you could stagger onwards by using GENJCL.USER to extract timestamps from RECON and build the appropriate commands/JCL for you. For example, if you wanted to recover a group of DBDS back to the their last image copies, you could use the following job to generate IBM recovery JCL from member ICLAST.

```
//PWAGENU JOB (1234),'P ARMSTRONG - UK',MSGCLASS=X,NOTIFY=PWA
//STEPl EXEC PGM=DSPURX00,REGION=OK
//STEPLIB DD DSN=IMSVS.DVT.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//IMS    DD DSN=IMSVS.DVT.DBDLIB,DISP=SHR
//JCLPDS DD DSN=PWA.SYSB.CNTL,DISP=SHR
//JCLOUT DD SYSOUT=(A,INTRDR)
//SYSIN DD *
 GENJCL.USER MEMBER(ICLAST) GROUP(ARMGRP) NOJOB LIST
/ *
```

Figure 48 Sample Job to Generate IBM Recovery JCL from ICLAST

Where ICLAST contains the following:

```
//PWAIC%GRPINDX JOB (1234),'P ARMSTRONG - UK',MSGCLASS=X,NOTIFY=PWA
//STEPl EXEC PGM=DSPURX00,REGION=OK
//STEPLIB  DD DSN=IMSVS.DVT.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//IMS    DD DSN=IMSVS.DVT.DBDLIB,DISP=SHR
//JCLPDS DD DSN=IMSVS.DVT.JCLLIB,DISP=SHR
//JCLOUT DD SYSOUT=(A,INTRDR)
//SYSIN DD *
%SELECT IC((%DBNAME,%DDNAME),LAST)
  GENJCL.RECOV DBD(%DBNAME) DDN(%DDNAME) RCVTIME(%ICTIME)
%ENDSEL
```

Figure 49 Sample JCL Contained in Member ICLAST

This will generate one recovery job for each member of the group. A much better solution is to use BMC Software’s RECOVERY PLUS or RECOVERY MANAGER for IMS and generate a single job for all the members of the group.

In this case ICLAST would contain the following:

```
%SELECT IC((%DBNAME,%DDNAME),LAST)
%DELETE (%STEPNO NE'OOOOO')
//PWAIC%GRPINDX JOB (1234),'P ARMSTRONG - UK',MSGCLASS=X,NOTIFY=PWA
//PWAREC%STPNO EXEC PGM=RVPUMAIN
//STEPLIB  DD
//SYSPRINT DD SYSOUT=*
//RVPSYSIN DD *
GLBL DBRC(Y) .....               global parameters
%ENDDEL
%DELETE (%GRPINDX NE 'l')
GROUP NAME(%DDNAME) .....        group level parameters
%ENDDEL
REC DBD(%DBNAME) TIMESTMP(%ICTIME) DDN(%DDNAME) one generated for each dbds
%ENDSEL
```

Figure 50 Sample JCL for Member ICLAST when Used with RECOVERY PLUS

Note the use of %GRPINDX in the above, incremented for each member of the group, and the use of %STEPNO to selectively generate global and group level information.

Example

Assume that an image copy (IC1) was made at T0. At T1 a log tape 1 (LT1) reflects the completion of batch job 1. Log tape 2 (LT2) reflects the completion of batch job 2. Change accumulation (CA1) is processed and includes these two log tapes. Then log tape 3 (LT3) reflects the completion of batch job 3. At this point, you realize that you mounted the wrong input tapes for batch job 2. You now want to recover the database to what it looked like after batch job 1 completed (T1). This is done with DBRC using a timestamp recovery. The timestamp to be used is T1. Issue a GENJCL.RECOV command specifying the database data set and the timestamp. The inputs to the recovery job are IC1 and LT1. Once this job is completed, the database is restored to the way it looked at T1.

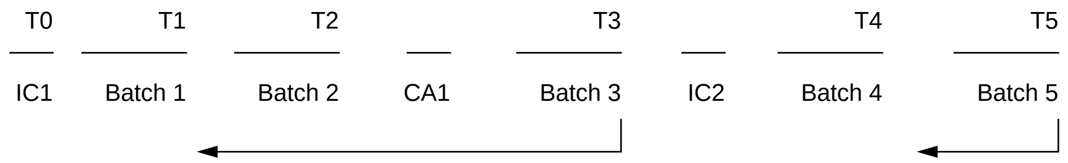


Figure 51 Selecting Timestamps for Recovery

Following this operation, assuming nothing was done but the timestamp recovery, an image copy (IC2) is processed. This image copy mirrors the database as it looked at T1.

Now batch job 4 and batch job 5 are run creating LT4 and LT5 respectively. Both did updates. At T5, it is determined that the input for job 5 was incorrect and a recovery to T4 is required. This situation is perfectly legitimate and can be handled by DBRC. Given this scenario, the inputs required for this example are IC2 and LT4.

Assume that the image copy at IC2 was not taken. What would be the inputs for the recovery to T4? By definition, timestamp recoveries require an image copy to make any subsequent recovery operations easier to perform. If IC2 was not taken, the required inputs to perform a timestamp recovery from T5 to T4 are IC1, LT1, and LT4. With these inputs, the database can be recovered to the way it looked at T4.

The effects of LT2 from batch job 2 and LT3 from batch job 3 effectively never happened due to the timestamp recovery taken at T3. However, the change accumulation in the middle of all this contains records that should never be used in a recovery. DBRC is supposed to realize this, but according to IBM RETAIN entries you *must* image copy after a timestamp recovery; otherwise, they cannot guarantee that DBRC will generate correct recovery JCL in the future. (If you use RECOVERY PLUS, you can generate an image copy and check the pointers in it as part of the recovery process.)

You can also use timestamp recovery to step through a series of logs (e.g., looking for a logic error in a program). You use the USEIC parameter for the first recovery and USEDBDS for any subsequent recoveries.

Note: When using USEDBDS, make sure you do not automatically generate a step to delete the database!

V5 Changes to Recovery Utility

Prior to IMS V5, the IMS utility always included complete logs. This means that is you used /DBR NOFEOV and then /STArted the database on the same OLDS, it was impossible to recover to the /DBR in the middle of the log with the standard IMS recovery utility. (BMC Software's RECOVERY PLUS can recover to a /DBR in the middle of a log even prior to V5—this is called special timestamp recovery—STR.) BMC Software's RECOVERY MANAGER for IMS can be used to find and create these recovery points.

As from IMS V5, the recovery utility was changed to only include the portions of logs required (in fact it uses the USIDs to determine this—see “USID” on page 163 for an explanation of USIDs). So now you can recover to a /DBR in the middle of a log and any log records after this will be ignored.

However, this code only exists in the recovery utility and not in the change accumulation utility, so if you ever want to run a recovery, which is only going to read a part of a log, you cannot use change accumulation, as change accumulation always includes *all* the records on the log. Fortunately, DBRC spots this and even though the change accumulation will run successfully and be recorded in RECON, at GENJCL.RECOV time, DBRC will realize that the change accumulation data set is useless and reselect the logs.

Warning! This is all wonderful until you have a case where you *have* to run change accumulation prior to recovery—e.g., n-way sharing (see also Chapter 17, “N-Way Sharing”). Now, you have the situation where you *cannot* recover the database with the standard utilities. Whoops! You should also study APAR No. II09694 for further details of what can go wrong.

The good news is that because RECOVERY PLUS sorts/merges any logs that it needs, this problem does not occur and recoveries to anywhere with any combination of logs or bits of logs is possible.

Non-Standard Recovery

If you decided to ignore all my advice and all generally perceived wisdom and used a non-standard image copy, and notified this to DBRC via the NOTIFY.UIC command, you will have to perform recovery as a three-step process:

1. First you recover the non-standard copy.

2. Issue the following command to inform DBRC when this restore is complete:

```
NOTIFY.RECOV DBD(...) DDN(...) RCVTIME(...) RUNTIME(...)
```

   RCVTIME is the timestamp of the user image copy recorded in RECON, which you have to go and look up and put in this command.

3. Now you apply any subsequent changes using GENJCL.RECOV with the USEDBDS parameter.

This is a tedious, error-prone procedure, recovery is a lot slower, and you should avoid it like the plague. If you use this, then I can't honestly believe that database integrity is your prime concern and I wouldn't bother to read the rest of this book!

Alternatives to Recovery

See Chapter 9, "Image Copy" for information on the use of MADS in Fast Path and/or the use of BMC Software's SECONDARY INDEX UTILITY/EP.

RECON Interaction on I/O Errors

I/O errors on Fast Path DEDBs and DL/1 databases cause interactions with DBRC, which are considered here.

Fast Path

DEDB I/O errors cause the CI in error to be deactivated (reads to this CI will cause an AO status code unless you have MADS and a good copy of the CI is available). Deactivated CIs are recorded in the second CI of the area. An error queue element (EQE) count is kept in RECON. See "Fast Path Considerations" on page 49 for a detailed description of Fast Path and DBRC.

DL/1 Database Write Error

If the subsystem with the write error invokes DBRC, the corresponding DBDS record is flagged (RECOVERY NEEDED). The RECOVERY NEEDED counter in the database record is incremented by one if this is the first write error on the data set.

When an I/O error occurs:

- The database in error is *not* stopped.
- An Extended Error Queue Element (EEQE) is created and logged—an EEQE is effectively the RBA of the broken block/CI.
- The EEQE is recorded in RECON—this means that DBRC knows which blocks/CIs in the database are broken.
- A DFS451 message is issued for the first error on each block/CI.

- For a read error
  - An AO status code is returned.
  - IMS retries the read of the block/CI for future requests.
- For a write error
  - No status code is returned, because the write to the database does not happen when you issue the call—it happens later.
  - A virtual buffer is allocated and the block/CI copied into it.
  - The RECOVERY NEEDED flag and counter are set in DBRC (further authorization is *not* denied).
  - The virtual buffer is logged.
- IMS retries the read/write when the database is closed.

IMS system restart rebuilds the virtual buffers unless, in the meantime, the database has been recovered. If the database has been recovered, the virtual buffer is discarded (no EEQE in RECON).

Authorization will be granted to other subsystems, except utilities that require a complete database (e.g., image copy and unload).

The /DIS DB command has a new keyword: BKERR. With this keyword you can display EEQE and incomplete backout information.

What this means is that you probably won't spot that you have had an I/O error (unless you have an MVS exit to trap the DFS451 message) until you come to copy or reorganize the database. You will now fail authorization. The action required is to recover the database and then continue with the copy/reorganization.

Chapter 12 BMC Software and DBRC

DBRC control is essential to the integrity and success of the reorganization process. DBRC manages the access to the database by granting varying degrees of authorization to the database or by prohibiting access. After a reorganization, all recovery resources relating to the disorganized version of the database are obsolete, and using them will damage the database. DBRC must be notified that a reorganization has taken place and when it took place, so that it will not try to use the obsolete resources for recovery.

The BMC Software reorganization utilities—UNLOAD PLUS® and UNLOAD PLUS®/EP, LOADPLUS® and LOADPLUS®/EP, and PREFIX RESOLUTION PLUS—use the same interfaces as the IMS utilities. They are much faster than the IBM utilities because they bypass DL/I and use track or cylinder I/O instead of block I/O. The EP utilities are faster than the "classic" BMC Software utilities, because they run parallel tasks.

Because the BMC Software unload and reload utilities contain exits that can manipulate the data during a reorganization, they also provide the fastest way for making significant changes to a database, such as alterations for year-2000 or Euro, deletion or archival of redundant data, etc.

The SECONDARY INDEX UTILITY, SECONDARY INDEX UTILITY/EP, and IMAGE COPY PLUS (ICP™) products have some features that have no equivalent in IMS. New methods have been developed to inform DBRC of the actions that are being performed.

The FAST REORG FACILITY (FRF™) and FAST REORG FACILITY/EP (FRF/EP™) products enable reorganizations of full-function databases to take place with concurrent read. CONCURRENT REORG for IMS enables reorganizations of full-function databases to take place with concurrent update.

The POINTER CHECKER PLUS® product is normally run under the control of image copy or IMAGE COPY PLUS and uses the authorization interface for image copies. However, when run in stand-alone mode with the DYNDB(IC) option it will interface with DBRC.

A complete description of the DBRC interface is given in the relevant BMC Software product manuals. This chapter is just an overview of the important bits.

The TRIMAR Fast Path utilities are covered in the next chapter.

Note: This chapter discusses BMC Software Extended Performance (EP™) utilities. In DBRC terms, the EP utilities essentially perform the same functions as the “classic” BMC Software Database Utilities that they replace, and I point out any differences, where relevant. For example, any discussion of LOADPLUS/EP is applicable to LOADPLUS unless otherwise noted.

UNLOAD PLUS/EP

UNLOAD PLUS/EP can perform the functions of HD unload utility and the scan utility, which request READ authorization.

Warning! No record is built in RECON at the end of unload. DBRC has no knowledge that the job has run. It is very easy to update the database between unload and reload. To avoid this potential corruption, it is recommended you issue the command

```
CHANGE.DB DBD(      ) NOAUTH
```

for each database involved in the reorganization prior to running any of the reorganization utilities. This command secures the database from use by any subsystem except for the reorganization and image copy utilities (SHARECTL only).

Note: The command does not affect any job currently running; it prevents any new jobs from starting.

The EP utilities actually check at reload time that no updates (ALLOCs) have occurred since the unload—if there have been updates, the reload will fail. The EP utilities also check for sensible things like not unloading empty databases, reloading full ones etc.

If any utility fails when you are running SHARECTL, the action required depends on whether the (E)STAE (cleanup) routines get control. With (E)STAE in control, rerun the job. A non-(E)STAE-RECON still contains a SUBSYS record plus the associated authorizations. You must issue the following command to do the cleanup, prior to restart.

```
CHANGE.SUBSYS SSID( ) ABNORMAL
```

UNLOAD PLUS/EP API

The UNLOAD PLUS/EP application program interface does not use DL/I. It accesses the database directly. Since the job has a PSB, DBRC performs authorization in the usual way.

In a non-sharing environment you get READ-ONLY or READ, depending on your PROCOPT. READ-ONLY shares with UPDATE and you are getting read without integrity. READ can only share with another READer so you get read with integrity.

In an n-way sharing environment, READ can share with UPDATE so your program may think it is getting read with integrity. However, because the API is accessing the database directly, you are actually getting read without integrity.

LOADPLUS/EP

LOADPLUS/EP requests EXCLUSIVE level authorization. After successful completion, it issues a NOTIFY.REORG to DBRC to write a REORG record in RECON. This also turns on the IMAGE COPY NEEDED flag in the DBDS record and increments the IMAGE COPY NEEDED counter in the database record. After reorganizing a database, image copy is required because the segments have all moved. DBRC does not allow recovery through a REORG record. Authorization is denied to all update subsystems until the image copy is run successfully.

If you define databases to DBRC as NONRECOV, LOADPLUS/EP and SECONDARY INDEX UTILITY/EP internally issue a NOTIFY.UIC command after the NOTIFY.REORG command if you specified NONRECOV(U) in your setup module. If you specified NONRECOV(N) in the setup module, then no DBRC commands will be issued.

LOADPLUS/EP LPI

The LOADPLUS/EP Load Program Interface lets you initially load an IMS database much faster than DL/I. The interface is transparent to the user load program. DBRC views it as a DL/I batch load job—EXCLUSIVE authorization, no log required.

The LOADPLUS/EP LPI issues a NOTIFY.REORG (and hence turns on the Image Copy Needed flag) at the end of a user load program. This is not done in any batch application using DL/I batch and a PROCOPT=L PSB (which I have always thought to be a black hole in DBRC). LOADPLUS/EP (quite rightly) considers issuing NOTIFY.REORG an enhancement over other IMS reload utilities and will force you to take an image copy after the user load program has finished.

Prefix Update—PREFIX RESOLUTION PLUS

Prefix resolution is a two-phase sort that has no interaction with DBRC. The prefix update portion of PREFIX RESOLUTION PLUS runs like a reload. At the end of the update procedure, it builds a REORG record in RECON and turns on the IMAGE COPY NEEDED flag and counter for each database updated. If the database has just been reloaded, DBRC builds one REORG record and sets the ICNEEDED information once.

PREFIX RESOLUTION PLUS allows you to split the prefix update work file and run the prefix updates in parallel. This is only possible with DBRC if you split the updates by physical database so that each prefix update job is updating a different physical database.

SECONDARY INDEX UTILITY/EP

SECONDARY INDEX UTILITY/EP can be used to recover secondary indexes and HIDAM primary indexes instead of using image copy and Recovery.

Registration Possibilities

There are three ways to use DBRC with secondary indexes:

- Register index and use image copy and recovery to recover.

- Register index and use SECONDARY INDEX UTILITY/EP to recover.

  Registering the index gives you protection if you are using SHARECTL, (for example, recovery forced after I/O error, backout forced after program abend, etc.). It also means that you can register all your databases and get to FORCER.

  In case 1, use the same rules as for your other databases. In case 2, you are registering the database for protection, but you will use SECONDARY INDEX UTILITY/EP to recover it if it is broken. Use the following command when registering the database.

  ```
  INIT.DBDS ........... GENMAX(2) NOREUSE
  ```

  Two is the minimum number of image copy generations allowed. As you are never going to image copy the database, there seems little point in specifying more (why you have to specify any at all is explained below). NOREUSE is required for the BMC Software DBRC interface to work.

  It is also recommended that you use the NONRECOV parameter, which defines the database as non-recoverable. See "RECOVABL/NONRECOV" on page 42 for further details.

- Do not register the index with DBRC.

  Not recommended at all (and why read this manual?). As I have said several times already (rather boring I know), you want to get all your databases registered and get to FORCER.

Implications of Registering — ALLOC Cleanup

When you define the GENMAX parameter in DBRC, it tells DBRC how many generations of image copy information to retain in RECON. When DBRC cycles through the number of generations you have specified, it performs a series of record cleanups in RECON.

Say you have specified GENMAX(2). When you take the third image copy, DBRC deletes the following types of records if they are older than the oldest image copy.

- ALLOC records
- LOGALL entries
- REORG, RECOV records

This allows you to delete any log records that no longer have active ALLOCations via the DELETE.LOG INACTIVE command.

If you are not image copying indexes because you intend to use SECONDARY INDEX UTILITY/EP to recover them and you are not using NONRECOV (what to do in this case is covered later), you must tell DBRC that image copies have been performed for the indexes. Otherwise the ALLOC/LOGALL/log cleanup procedure does not work, and your RECONs will grow and grow. When you image copy the main database, issue the NOTIFY.UIC command or NOTIFY.IC commands for the indexes.

NONRECOV

You are recommended to define the indexes as NONRECOV, NOREUSE, and GENMAX(2), and issue two NOTIFY.IC or UIC commands. Since ALLOC or LOGALL records are never created, you don't have to issue these commands again.

Note: To convert an already registered database to NONRECOV, the sensible option is to delete the database from RECON (this also deletes ALLOC, LOGALL, IC, REORG, RECOV, CAGRP entries, DBDSGRP entries) and then re-register the database and ADD it to the relevant DBDSGRPs. Using the CHANGE.DB NONRECOV command throws DBRC and you into utter confusion—don't use it.

Note: Someone once told me that you cannot block-level share NONRECOV databases—this is not true, it works fine.

DBRC Interface

The DBRC interface is slightly different in SECONDARY INDEX UTILITY and SECONDARY INDEX UTILITY/EP.

SECONDARY INDEX UTILITY

There are two parts to SECONDARY INDEX UTILITY—the scan function and the load function. Scan uses DL/I, or if you have UNLOAD PLUS, then you can use UNLOAD PLUS to perform the scan (at a much higher speed). When the load function completes successfully, it may issue a series of commands to DBRC.

SECONDARY INDEX UTILITY/EP

When the scan function is used, the primary database requires READ authorization from DBRC. Regardless of whether the scan function is used, the index databases require EXCLUSIVE authorization.

In SECONDARY INDEX UTILITY/EP only, there is a setup option called NONRECOV. If you specify NONRECOV(N), then no DBRC commands are issued for indexes that are defined to DBRC as NONRECOV. If you specify NONRECOV(U), or if you are using “classic” SECONDARY INDEX UTILITY, the following commands are issued:

1. CHANGE.DBDS NORECOV

   This turns off the RECOVERY NEEDED flag. This is turned on if the database experienced an I/O error.

2. NOTIFY.REORG

   Builds the REORG record.

3. ICNEEDED(OFF) - NOTIFY.UIC

   Selecting this option in SECONDARY INDEX UTILITY/EP issues a NOTIFY.UIC command for the database in question. The IMAGE COPY NEEDED flag is turned off. Use this with NONRECOV.

   OR

   ICNEEDED(ON) - RUN IC/IC+

   Select this option to image copy the database after building it, creating it, rebuilding it. Use this with RECOV.

   SECONDARY INDEX UTILITY/EP checks for your DBRC setting (RECOV or NONRECOV) and sets its default accordingly:

RECOV	ICNEEDED(ON)
NONRECOV	ICNEEDED(OFF)

FAST REORG FACILITY/EP

BMC Software recommends the use of DBRC for FAST REORG FACILITY/EP (FRF/EP).

FRF/EP reads the source database and creates a reorganized shadow database. It then swaps the two (or you can do this process manually). The source database is the one registered in RECON—the target database is controlled by FRF/EP and is not registered in RECON. If you specify ICP(YES) to invoke the Parallel Image Copy feature, FRF/EP calls IMAGE COPY PLUS to take the post-reorganization image copy automatically as the database is written. You can also run BMC Software's POINTER CHECKER PLUS under IMAGE COPY PLUS to check pointers while IMAGE COPY PLUS is running under FRF/EP.

If you execute a standard FRF/EP reorganization and you discover any errors before the swap of data set names, abandon the reorganization and delete the output files. If DBRC has not been notified of the reorganization, issue the CHANGE.DB command to allow authorization requests. However, if DBRC has already been notified of the reorganization, the ICNEEDED flag in DBRC may be turned on.

You can delete the reorganization record and the image copy record from the RECON data sets. The DBD is still associated with the data sets it has had all along—the disorganized ones.

FRF/EP manages a reorganization in one step (it passes the work files from UNLOAD PLUS/EP to LOADPLUS/EP and from LOADPLUS/EP to SECONDARY INDEX UTILITY/EP directly in storage). Because DBRC authorization is obtained at the beginning of the reorganization and released at the end, no other batch access can be obtained between the unload and the reload.

CONCURRENT REORG

CONCURRENT REORG reorganizes a database using FRF/EP. At the same time it captures the changes to the source database and applies these to the shadow as well. At the end it swaps the two.

CONCURRENT REORG requires full DBRC registration, and DBRC must be active. When you reorganize a database with CONCURRENT REORG, FRF/EP completes its normal DBRC-related tasks. CHANGE RECORDING FACILITY performs additional DBRC-related tasks at the beginning of the reorganization and after the change apply process completes. All DBRC activities (and the data set name swapping process) are delayed until the change apply process is finished.

CHANGE RECORDING FACILITY creates a file that contains the updates made to the database while it is being reorganized. After the database is reorganized, a batch update program applies these updates to the reorganized database. This batch apply program creates a log, which CHANGE RECORDING FACILITY registers to DBRC. To correlate the database to this log, CHANGE RECORDING FACILITY creates allocation records.

If you use the image copy that was created during the CHANGE RECORDING FACILITY reorganization to recover the database, the CHANGE RECORDING FACILITY batch log should be included. Although the image copy is technically intact (it has no pointer errors), the user data is not complete until the CHANGE RECORDING FACILITY batch log is applied.

BMPs

When using FRF/EP or CONCURRENT REORG, you are probably running reorganizations while IMS/DBCTL is running. Hence, there may be BMPs running against the database. To avoid waiting for these to complete FRF/EP and CONCURRENT REORG have an interface to BMC Software's APPLICATION RESTART CONTROL product, which has the facility to "pause" a BMP and restart it. This means that you don't have to wait for long-running BMPs to finish before you run a reorganization or swap the source and target.

Common Errors

Common errors in the reorganization process are to forget to update the production libraries, e.g., ACBLIB with the new DBD, or to load a production database using the test DBD. Online IMS happily opens your reorganized database with the wrong DBD and promptly corrupts it. Frequently you will first notice this hours later when you start getting duplicate records, missing records etc. Correcting databases corrupted in this way can take many hours and often involves coding special programs, hand-merging records etc.

BMC Software's DATABASE INTEGRITY PLUS avoids this problem by ensuring that a database is opened with the same control blocks as were used during the load/reload. If this check fails, you will receive a DBRC authorization failure and the database will not be corrupted. You are not recommended to put the "labels" used by DATABASE INTEGRITY PLUS in the RECONs, as the DBRC UPGRADE process gets upset by record types that it does not recognize.

RECOVERY PLUS and SECONDARY INDEX UTILITY/EP

The traditional method for recovering an index is to use the same procedures that you would use when recovering any non-index database. You would restore the index using a previous image copy and then apply the appropriate change records.

The second method implicitly rebuilds all associated primary and secondary indexes while the primary database is being recovered. Specify BLDINDEX(Y) to rebuild primary and secondary indexes for the named DBD.

The third method uses the SIUDSN keyword. When the data set name of the input image copy matches the value of the SIUDSN keyword, RECOVERY PLUS automatically invokes the SECONDARY INDEX UTILITY/EP to rebuild secondary indexes without recovering the primary database.

If you define your indexes as non-recoverable, RECOVERY PLUS automatically invokes SECONDARY INDEX UTILITY/EP to rebuild them under either of the following conditions:

- You specify BLDINDEX(Y).
- You explicitly code the AREC or REC control statement for the index database and the image copy data set name matches the SIUDSN value.

Whenever you image copy the main database, issue a NOTIFY.IC command for the indexes:

```
NOTIFY.IC DBD(SIDBD1) -
          DDN(SIDBD1P) -
          ICDSN(BMC.SIUDUMMY.DSN) -
          VOLLIST(DUMMY) -
          RUNTIME(YYDDDHHMMSST)
```

Figure 52 Sample NOTIFY.IC Command for Indexes

IMAGE COPY PLUS has a facility—Virtual Image Copy—which will perform these notifications for you when you copy the main database.

With NONRECOVerable databases, it is theoretically unnecessary to perform any more NOTIFY.ICs after the initial two at registration time. However, you are recommended to issue NOTIFY.IC whenever you image copy or reorganize the main database. This is for two reasons:

- You now have a record in RECON that the indexes were not “forgotten.”
- It keeps your RECONs “up-to-date.”

IMAGE COPY PLUS AOI and DBRC

You can use this interface to issue commands in your job stream, (for example, /DBD and /DBR).

IMAGE COPY PLUS determines the level of command required. For recovery, it issues a /DBR. For image copies, it issues a /DBD. These commands build a DEALLOC timestamp in the ALLOC record in RECON and flush the buffers. You can now run a batch image copy, online image copy, etc.

IMAGE COPY PLUS verifies that the database is deallocated using DBRC. This is actually done by testing for authorization. The level of authorization that IMAGE COPY PLUS tests for is shown in the next section.

IMAGE COPY PLUS Authorization Levels

The levels requested from DBRC are:

- batch image copy—READ
- incremental image copy—READ ONLY
- copy image copy—READ ONLY
- online image copy—READ ONLY

Online image copy reads the database while it is being updated. DBRC calls this database level sharing. The prerequisites to this are:

- DBRC turned on
- ACCESS <= UP online
- DISP=SHR
- SHRLEVEL > 0
- VSAM SHAREOPTIONS >= 2

Many people use database level sharing with batch image copy. They issue a /DBD command, and run a batch image copy job in parallel with online READ transactions.

RECOVERY PLUS

RECOVERY PLUS uses the same interfaces as the IBM recovery utility. It requires exclusive use of the database. Any subsystems that are currently authorized against the database must have terminated abnormally. After successful recovery, the RECOVERY NEEDED flag and counter are reset. If recovery fails, the RECOVERY NEEDED flag is left on to protect the database.

RECOVERY PLUS can also image copy the database(s), notify DBRC of the image copies, invoke POINTER CHECKER PLUS or FAST PATH ANALYZER/EP to verify the integrity of the database(s), and invoke SECONDARY INDEX UTILITY/EP to restore the primary and secondary indexes.

The BMC Software manuals contain examples of how to use IMAGE COPY PLUS and RECOVERY PLUS to copy and restore groups of databases, (for example, copying and restoring a pack).

RECOVERY PLUS supports DL/I, DEDB, and MADS. It also allows you to recover a database to *any* point in time. This unique feature allows you to recover databases to any desired point, and does not apply any of the in-flight transactions, so no backout or cleanup is required. This also solves recovery problems in n-way sharing—see Chapter 17, “N-Way Sharing” and the BMC Software manual *IMS DB Backup and Recovery With N-Way Sharing* due out shortly.

RECOVERY MANAGER for IMS (RMGR)

The RECOVERY MANAGER for IMS (RMGR™) product estimates the time required to recover, ensuring synchronization of the recovery across databases, establishing valid recovery points, and ensuring that the inputs necessary for recovery are available. With RMGR and a scheduling utility, you can schedule log analysis and the creation of planned recovery points. RMGR allows you to perform an automatic recovery once you have analyzed your recovery assets and selected a recovery point.

Group processing enables you to manage and synchronize backups and recoveries across applications and databases. You can create groups based on DBD, PSB, CAGRP, DBDSGRP, VOLUME, SUBSYS, and DSN. You can also define profiles to specify the utilities you want to use. RMGR supports BMC Software and IMS backup and recovery utilities. RMGR supports full-function databases and DEDBs.

RMGR supports a data-sharing environment where two or more IMS online subsystems are sharing databases (use the SHAREGROUP keyword).

You can recover a group of objects to the following recovery points:

- current
- last batch image copy for each object
- last batch image copy prior to a specific time stamp
- any point-in-time (using the RECOVERY PLUS PIT function)
- common recovery point determined by the find recovery points function

Using RMGR, you can request that additional recovery points be found. You can generate points of consistency for groups and also identify quiet points for an object or a group of objects.

When you request the Hold Point of Consistency (HPC) function, the objects in the group are quiesced and are held in this state until the process terminates or until a user-specified time-out value expires.

When you request a recovery assets check, all assets necessary for a recovery to current are checked. The existence of primary or secondary image copy data sets, primary or secondary input log data sets, and change accumulation data sets is verified. RMGR performs a GENJCL and reads the RECON data sets to create the inputs to the recovery assets check. The internal GENJCL is generated even though the online logs are open and the databases are allocated.

RMGR provides a positive-response IMS command batch interface that you can use for all utility backups and recoveries. This interface allows you to create sets of commands to schedule batch IMS jobs with the assurance that the jobs will not attempt to run until the databases are actually closed or opened.

Chapter 13   Fast Path Utilities

The BMC Software TRIMAR utilities perform a variety of functions for Fast Path users. Their interaction with DBRC is considered here, firstly for the standard TRIMAR utilities and then for the FAST PATH/EP series.

TRIMAR FAST RECOVERY UTILITY

The TRIMAR FAST RECOVERY UTILITY product provides an extremely fast method of recovering all DEDBs and MSDBs with complete integrity, allowing a cold start of the IMS/VS system to be performed. It is intended to be used in the following scenarios:

- emergency restart failure (also see “/ERE Failure” on page 98)
- when a cold start is more expedient than emergency restart

This utility automatically selects all information and resources required for recovery and performs the recovery functions.

The DBRC Interface

TRIMAR FAST RECOVERY UTILITY reads the RECONs to determine which log data sets are required for recovery. Normally, you should use dynamic allocation of the RECONs. If you wish to override DBRC’s RECON selection mechanism, you can specify a RECONUSE DD statement to point at a specific RECON. If you do not wish to use DBRC, code DBRC=NO and supply the SLDS, OLDS information using DD statements. You must close the OLDS using DFSULTR0 or RCF (see below).

If you have not generated a list of PSBs needing backout from the OLDS closure, specify PSBLIST=YES on the CTL control statement of TRIMAR FAST RECOVERY UTILITY to generate a list. You should also specify a DBRCCTL DD statement. TRIMAR FAST RECOVERY UTILITY generates and executes DBRC commands to turn on the BACKOUT NEEDED flags for the registered databases referenced by the listed PSBs. This prevents these databases from being used by the online system until batch backout is successfully performed.

For DBRC Share Control customers, the RECOV NEEDED flag is automatically turned on for registered areas so the subsequent forward recoveries can be run.

The Recovery Step

The second step recovers the DEDBs. Supply a DBRCCTL DD statement in this step. If this step fails to recover an area for any reason (e.g., HSSP Image Copy was running), appropriate DBRC commands are generated and executed to protect the area.

TRIMAR RESTART CONTROL FACILITY

The TRIMAR RESTART CONTROL FACILITY (RCF) product is a comprehensive, automated mechanism that ensures the integrity of IMS databases and critical system data sets during the restart of an IMS control region.

RCF is a standard OS job step that precedes the control region step in the IMS job stream or started-task JCL procedure. It assesses the current state of the IMS system and performs any required recovery activities.

If IMS is started with AUTO=YES, RCF performs the following functions:

- Ensures that MSDB integrity can be maintained. If MSDB integrity will be compromised by an AUTO=YES restart, RCF will automatically take corrective action.
- Ensures that the RDS is usable by IMS.
- Ensures that the set of MSDB checkpoint and RDS data sets used by the surveillance system are present and usable in an IMS-XRF environment.

When IMS is started with AUTO=NO, RCF determines if the IMS system is a partner in an IMS-XRF active complex. If so, and this is a restart for the surveillance system, a standard restart of that system will be performed. If DBRC indicates the surveillance system is active, DBRC is notified it is no longer active.

If the restart is for a non-XRF system or for the active partner system in an IMS-XRF complex, then RCF automatically performs the following tasks:

- Invokes DFSULTR0 to close the last OLDS, if required.
- Invokes DBRC to indicate IMS abnormally terminated, if required. This removes the necessity to specify OVERRIDE in the IMS restart command.
- Deletes, reallocates, and restores the MSDBCPn data set from the best source (the other MSDBCPn data set, MSDBDUMP, or a backup data set) if an MSDB checkpoint data set is flagged as unusable.
- Invokes DBFDBDR0 to create an up-to-date MSDBINIT data set. If IMS terminated normally, an unload is performed. If IMS terminated abnormally, the RECONs are interrogated to determine the log data sets needed, and DBFDBDR0 is invoked to perform a RECOVER function.
- Selects the best IMS restart command or validates a user-entered restart command and passes it to the IMS control region.
- Optionally, invokes TRIMAR FAST RECOVERY UTILITY to recover all DEDBs, and schedules any required full-function database backouts.

FAST PATH/EP Series

To help maintain DEDBs, BMC Software created FAST PATH/EP Series products to provide more efficient and effective solutions to DEDB analysis, performance, space management, and capacity management. The FAST PATH/EP Series includes the following products and functions:

FAST PATH ANALYZER/EP™

- analyzes information about single or multiple areas
- validates pointer data
- generates reports
- retains statistical data in a repository for historical reporting
- analyzes either an area data set or an image copy data set

FAST PATH ANALYZER/EP Online™

- performs the same analysis functions as FAST PATH ANALYZER/EP while leaving the DEDB online
- extracts data from an online DEDB

FAST PATH REORG/EP™

- reorganizes single or multiple areas
- performs space reclaim maintenance
- reorganizes only UOWs that require reorganization
- can be used to extend IOVF and SDEP space allocations
- can be used to restructure the DEDB
- creates an output image copy during processing
- provides high-performance unload/reload processing capabilities

FAST PATH REORG/EP Online™

Performs the same space reclaim maintenance as FAST PATH REORG/EP while leaving the DEDB online

FAST PATH ONLINE IMAGE COPY/EP™

Performs an image copy while leaving the DEDB online

These FAST PATH/EP Series products directly replace several Trimar Fast Path Series products—they are faster, allow more parallelism of tasks, offer new functions and have greatly simplified JCL.

EP Product	**Replaces**
FAST PATH REORG/EP	TRIMAR FAST PATH UNLOAD/RELOAD
FAST PATH ANALYZER/EP	TRIMAR FAST PATH ANALYZER
FAST PATH ANALYZER/EP Online	TRIMAR FAST PATH ONLINE UTILITIES
FAST PATH ONLINE IMAGE COPY/EP	TRIMAR ONLINE DEDB IMAGE COPY

Figure 53 FAST PATH/EP Products Replacing Trimar Products

DBRC Function Used by FAST PATH/EP Series Products

When executing in offline mode, the FAST PATH/EP Series products can interface with DBRC to perform several activities. The use of DBRC is optional, but recommended.

When DBRC is active during the execution of FAST PATH/EP Series product execution, it is used for the following functions:

- access control—The FAST PATH/EP Series products request access authorization for each area prior to its use. When the product is finished processing the area, it requests unauthorization of the area. The authorization level used for the area depends on the requirements of the product function that is executing.

- dynamic allocation of area data sets—The FAST PATH/EP Series products use the registered name of the area data sets for dynamic allocation. If multiple area data sets (MADS) are specified for an area, then FAST PATH/EP uses the first data set in the ADS list.

- recovery control—The FAST PATH/EP Series products record events, as necessary, that cause a change in the recovery status of the area.

- image copy notification—The FAST PATH/EP Series products record the successful creation of image copy data sets.

FAST PATH ANALYZER/EP (and Online)

When running offline, FAST PATH ANALYZER/EP requests READ authorization, or READ-GO when running in concurrent mode (runs in batch with database online). When running in online mode, it runs as a Fast Path region.

You can also invoke FAST PATH ANALYZER/EP from within BMC Software's IMAGE COPY PLUS and RECOVERY PLUS products—see Chapter 12, "BMC Software and DBRC". This capability allows the analysis and utility functions to share the database I/O.

FAST PATH REORG/EP (and Online)

When running offline, FAST PATH REORG/EP requests EXCLUSIVE authorization. When running in online mode it runs as a Fast Path region.

When an offline reorganization is completed, FAST PATH REORG/EP informs DBRC that an image copy is required for the area by setting the IMAGE COPY NEEDED flag.

When DBRC is active during the execution of the change function command, and the area is registered with DBRC, FAST PATH REORG/EP verifies from DBRC that each output area is marked as RECOVERY NEEDED. It also obtains authorization for exclusive access (EX) from DBRC for each output area data set. No DBRC authorization is performed on the input data sets.

You can use the IC subcommand to request that an image copy be created for each area during the reorganization or change function. When the image copy is created successfully, a NOTIFY.IC command is issued to register the image copy in DBRC and the IMAGE COPY NEEDED flag is reset.

If you do not take an image copy during the offline reorganization process, a warning message is issued indicating that an image copy needs to be taken for each area.

Note: BMC Software recommends that you ensure that a valid image copy of the input area exists before executing the reorganization function in offline mode or the change function. Also, in offline mode, you must take an image copy of the output area during or after executing the REORGANIZE command or change function.

When using the unload function, READ authorization is requested. When DBRC is active during the execution of the reload function, and the output area is registered with DBRC, FAST PATH REORG/EP verifies from DBRC that each output area is marked as RECOVERY NEEDED.

The reload function obtains exclusive access (EX) from DBRC for each output data set. No DBRC authorization is performed on the input data sets.

At the successful completion of the reload function for each output area, the RECOVERY NEEDED flag is automatically reset and the IMAGE COPY NEEDED flag is set indicating that an image copy is required.

If your areas do not reflect the appropriate RECOVERY NEEDED condition, then use the following command:

```
CHANGE.DBDS DBD(dbdname) AREA (areaname1) RECOV
```

When DBRC is active during the execution of the initialization function, and the area is registered with DBRC, FAST PATH REORG/EP verifies from DBRC that the area is marked as RECOVERY NEEDED. It also obtains authorization for exclusive access (EX) from DBRC for the area data set.At the completion of the initialization function, the RECOVERY NEEDED flag is automatically reset for the area.

Image Copy Options

Use the image copy function to generate one or more image copies of a DEDB. The IMAGECOPY command executes the FAST PATH ONLINE IMAGE COPY/EP product. It can be executed in the online environment only.

Many FAST PATH/EP Series product functions provide the facility to create an image copy of the DEDB at the same time. An image copy is requested by using the IC subcommand.

Specifying the IC subcommand with a function in online mode creates an image copy by using the FAST PATH ONLINE IMAGE COPY/EP product. Specifying the IC subcommand with a function in offline mode requires the IMAGE COPY PLUS product.

If DBRC is active, and if the area being processed is registered within DBRC, the image copy data set or data sets produced will be recorded using the NOTIFY.IC process. There is no limit to the number of image copy data sets you can create, but no more than two image copy data sets can be recorded within DBRC. You can control which image copy data set or data sets will be recorded by using the NOTIFY keyword.

Chapter 14 IBM Utilities

Do not use the Utility Control Facility (UCF) together with DBRC—the results are unpredictable.

Unload and Scan Utilities

Hierarchic direct (HD) unload requests READ authorization at initialization time. This is because HD unload can unload only one database per execution. On the other hand, hierarchic sequential (HS) unload or scan can unload or scan multiple databases during one execution through the use of multiple control cards. In this case, authorization is not obtained at initialization, but as required.

Reload Utilities

The reload utilities request EXCLUSIVE level authorization. The reload jobs are not considered as update jobs by DBRC. This means that a log is not needed and an ALLOC record is not created by the reload utility.

When the HD reload utility completes, it builds a REORG record in the RECON data set. This is used to tell DBRC not to recover across this reorganization. Additionally, the IMAGE COPY NEEDED flag is turned on. The correct way to turn this flag off is to perform image copy.

HISAM reload operation depends on how the database data set has been registered in RECON. A REORG record is always built in the RECON data set for the same reasons that HD reload builds this record. Because the input to HISAM reload (i.e., the HISAM Unload) can be used instead of an image copy in the database recovery utility, DBRC tries to tell this to RECON.

If the database data set is registered as NOREUSE, when the reload is complete, DBRC builds an image copy record in the RECON data set with a timestamp of 0.1 second later than the REORG record. The IMAGE COPY NEEDED flag is off.

However, if the database data set is registered as REUSE, DBRC cannot record the HISAM unload because it is not one of the predefined data sets entered with an INIT.IC command. The IMAGE COPY NEEDED flag is turned on.

Prefix Update without a Log

Prefix update can run in two different ways—with a log or without a log. When running without a log, EXCLUSIVE authorization is requested. Although it updates the database, it is not considered to be an update job for the same reasons that the reload utilities are not considered to be update jobs.

When prefix update completes processing, it builds a REORG record for all the updated databases and turns on the IMAGE COPY NEEDED flag. Since the databases are updated, logs reflecting these updates do not exist. A new recovery point needs to be established for the database.

Prefix Update with a Log

EXCLUSIVE authorization is still required. However, since this utility is running with a log, DBRC treats this job as a batch update job. This means the prefix update utility, running with a log, creates log records and an ALLOC record. Since it is processed as a batch job, the REORG records are not built and IMAGE COPY NEEDED flags are not turned on, as they are not needed. I’ve always thought this is a pretty stupid way to run prefix update, as it will be slower and needs backout if it goes wrong, but no doubt someone will ring me up and explain why it is the best thing since sliced bread!

Utility Failure

If a utility fails and the (E)STAE routines get control, the RECON records are cleaned up and you must rerun the utility. If running prefix update with a log, backout is required.

If the (E)STAE routines do not get control, a SUSBYS record that needs clearing is in RECON. Use the CHANGE.SUBSYS ABNORMAL command to do this for all DL/I utilities except backout and prefix update. If running backout, simply rerun. If running prefix update with a log, close the log and run backout.

Warning! Remember that this command cannot tell the difference between a non-(E)STAE failed subsystem and one that is currently running, so only use it when you are sure you know what you are doing.

Chapter 15 IMS and the TOD Clock

The basic principle of IMS forward recovery is the application of change records to a previously recorded image copy of the database data set. To achieve this accurately, log records and image copies are timestamped. However, this in turn means that the TOD clock(s) in the CEC(s) must be set accurately.

This chapter considers the implications that changes to and inaccuracies in the TOD clock have on the IMS forward recovery utilities in both a single and a multi-CEC environment.

The code seems to change back and forth between releases, but this is how I believe it should be, and it is based on the only explanation I have ever been able to find of DSSN, USID, and Merge Needed—the ETR numbered RTA000096319.

IMS Use of Timestamps

IMS stores timestamps in the log records, the headers of image copy data sets and in the RECONs. It also uses two other fields—DSSN and USID to work out when logs need sorting—i.e., when a *merge needed* is required.

DSSN

The DSSN is used by DBRC to help determine when a data set is being shared and when a merge operation on the logs is required prior to a database recovery.

The DSSN for a database data set or DEDB area appears in three places.These are the RECON DBDS (area recovery) record, the RECON ALLOC record, and the IMS database change records. These three values may or may not always be the same.

DSSNs are maintained when the first update following database (area) open is performed by a subsystem (i.e., when an ALLOC record is to be created for a subsystem). The logic used for DSSN maintenance is as follows:

- If an open ALLOC record is found for another subsystem (open ALLOC is defined as an ALLOC with a zero stop time and its subsystem is still executing), then do not increment the DSSN in the RECON DBDS (area recovery) record and use the DSSN found in the open ALLOC record as the DSSN to be used for the new ALLOC record, as well as for the IMS database change records.

  This is the situation when a database data set or area is truly being updated by multiple subsystems at the same time (n-way sharing).

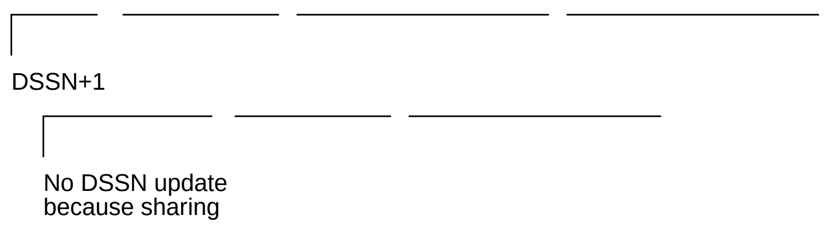


Figure 54 DSSN in N-Way Sharing

- If a closed ALLOC record is found for another subsystem and the log data set (OLDS or archived PRILOG data set entry) that was in use at the time the found ALLOC record was closed had not been closed, then use the DSSN found in the closed ALLOC record as the DSSN in the ALLOC record being built. However, increment the DSSN in the RECON DBDS (area recovery) record and use the incremented value in the IMS database change records.

  The database is taken offline without closing the online log and then updated in batch. Non-sharing, but concurrently created logs, now require merging.

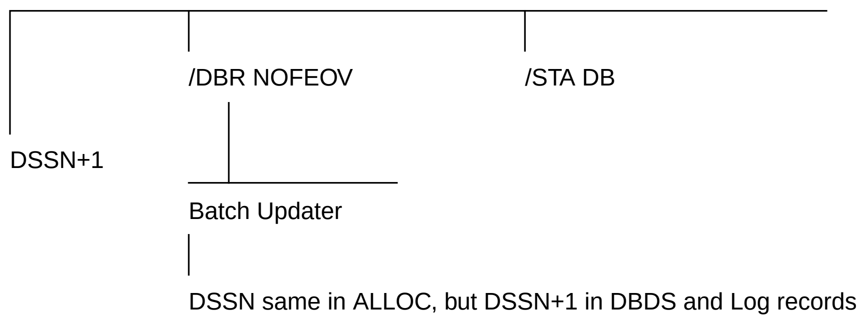


Figure 55 Incrementing DSSN after /DBR NOFEOV

- If neither of the above situations occurred, then increment the DSSN in the RECON DBDS (area recovery) record and use the incremented value in both the ALLOC record being built and the IMS database change records.

  The database is updated by two separate non-concurrent subsystems.

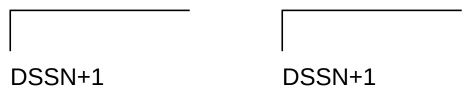


Figure 56 Incrementing DSSN in a Non-Sharing Environment

The change log records are merged (when needed—see below) as part of the change accumulation or RECOVERY PLUS execution, and DSSN is used as one of the sort fields to make sure that the records are all in the right order.

It is distinctly possible that the code does not work exactly like this in your shop, because I find APARs on this almost every month—for instance, here is an extract from a new APAR PQ15667 concerning a bug corrected for DSSN processing:

At one time it was thought that when two or more logs required merging (i.e., one log that spans another log and contains changes both before and after the changes on the middle log) then all ALLOCs within the merge scope had to have the same DSSN. This is not the case since other DBRC code determines a merge environment without the benefit of the DSSN values. So all of the code in DSPALD00 that determined merging has been removed. Now, ALLOC records will be assigned a common DSSN value only when two or more subsystems are sharing the database.

The moral here is that DBRC will work out if a merge is needed or not. If you don't use DBRC (silly person), then I suggest you always run change accumulation prior to recovery, or better and safer use BMC Software's RECOVERY PLUS.

Armstrong’s Confusion

Now, my problem is that I learned many years ago that this code was also designed to help you in the case where you were running jobs on two machines, which did not have exactly the same TOD clock (because in those days we did not have sysplex timers, and nowadays not everybody has bought one yet). This means that you could have jobs recorded in RECON in a different order to the one in which they actually ran.

For example, if MVSA thinks it is Monday and MVSB thinks it is Tuesday (actually they may only be a few seconds out, but this is easier on my brain and my typing):

- You run Job 1 on MVSA—it gets recorded in RECON as Monday 9:00 say with a DSSN of 1.

- You run Job 2 on MVSB—it gets recorded in RECON as Tuesday at 9:05 say with a DSSN of 2.

- You run Job 3 on MVSA—it gets recorded in RECON as Monday at 9:10 say with a DSSN of 3.

I tried this and much to my surprise, rather than taking it in its stride and simply sorting the log records using DSSN, DBRC came over all nasty and gave me the following messages:

```
DSP0246I DSSN PROCESSING UNABLE TO CONTINUE
DSP0246I DBDNAME=ABCD1234 DDNAME=ABCD1234 DSSN=00000003
DSP0209I PROCESSING TERMINATED WITH CONDITION CODE = 12
```

I suppose the moral of all this is to buy a sysplex timer, or run all your jobs on one MVS. If you ever get these messages, I suggest you check your clocks, stop IMS work, and take some fresh image copies.

Note: If anyone has read this far and has got any thoughts/comments/ nostalgic reminiscences, feel free to send me an email.

One customer managed to IPL and run the systems as shown and described below.

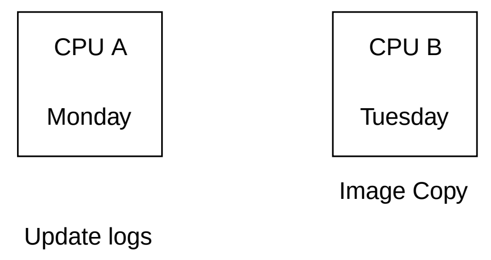


Figure 57 The Dangers of Different TOD Clocks

In this scenario DBRC will:

1. Record the image copy in RECON as Tuesday.
2. Record later logs as Monday.
3. Reset the database to image copy status and lose log updates with the recovery utility.

This is, in fact, an IMS problem as well as a DBRC problem, since recovery and change accumulation reject change records based on the timestamp, change accumulation uses the purge time of latest image copy, and recovery discards records earlier than the image copy header. There is nothing in DBRC to prevent this situation. Another reason for a sysplex timer?

USID

The USID was added in IMS V5 for the Remote Site Recovery feature, but is active in all environments. One benefit that the USID logic provides the non-RSR user in an IMS V5 environment is that it allows the IMS Recovery utility to stop applying updates from logs prior to hitting EOF on the logs. In other words, this is the code that the IMS recovery utility uses to determine which log records should be included from a log and which not.

The USID for a full-function database is stored in the RECON DB record and is also stored in the IMS database data set change records. The USID for a DEDB area is stored in the RECON area authorization record, as well as the IMS change records.

USIDs are maintained when the first update following database open is performed by a subsystem (i.e., when an ALLOC record is to be created for a subsystem). The logic used for USID maintenance is as follows:

- Full-Function

  If current value of USID is *0*, make it *2* and return the value to IMS for use in the change records. By definition, this can only occur if no other subsystem is currently using the database. I have no idea why *1* was skipped. This is the way the code is written.

  If any other currently running subsystem has an open ALLOC record against any database data set within the physical database, do not increment the USID. Return the current value to IMS for use in the change records.

  If no other currently running subsystem has an open ALLOC record against *any* database data set within the physical database, increment the USID in the RECON DB record and return the incremented value to IMS for use in the change records.

- DEDBs

  Set the USID to the DSSN + 1 (DSSN maintenance for DEDB areas occurs prior to USID maintenance), and return the value to IMS for use in the change records. Note that the USID for a DEDB area will always be *1* greater than the DSSN.

Merge Needed

This section will describe how DBRC determines if a *merge needed* condition exists.

There are two types of merge needed conditions within DBRC. These are *true merge* and *required merge* and they apply to log input to the IMS database recovery utility.

True merge represents the situation where two or more different IMS subsystems are concurrently updating the same database data set or area (block-level data sharing). This can be identified, from a RECON record perspective, by the occurrence of overlapping ALLOC records that point to log data sets that have not been processed by change accumulation.

Required merge is a more subtle situation. DBRC detects this by the following logic:

- Order the ALLOC records in allocation time sequence and assign consecutive numbers to the ALLOC records.

- Obtain the log volume stop time for the last log volume contained in each of the ALLOC records (i.e., the stop time for the log volume in use at the ALLOC deallocation time), and store it with the ALLOC record.
- Sort the ALLOC records in log volume stop time sequence.
- Check the ALLOC records to see if they are still in allocation time sequence (i.e., check to see if the sequence numbers assigned in step 1 are still in sequence). If they are not, then a *required merge* merge needed situation exists.

The required merge situation logic is based on DBRC's insistence that the input logs to the recovery utility be ordered in end of volume time sequence, even if it doesn't make sense.

Image Copy

The image copy (IC, OLIC, and CIC) output data set has a timestamp in the header that is the timestamp of the start of the job. The recovery utility checks this timestamp against the timestamp stored in RECON. This was introduced to prevent the mistake of inputting the wrong image copy to recovery, e.g., wrong GDG generation. It has caused problems with some customers though, as there are some procedures in practice that take an image copy at one point in time and NOTIFY it a later point to the RECONs, e.g., using XSA/Snapshot to “snap” a volume and then running an image copy from this snap.

CONCURRENT REORG also creates this problem, but it provides a solution when using the IMS recovery utility, and BMC Software’s RECOVERY PLUS can handle this situation.

Change Accumulation Purge Time

This is the latest image copy start time or latest change accumulation execution time for this group for batch image copies. For online copies, it is the previous ALLOC time. It is used to discard any log change records earlier than this timestamp.

Recovery

This utility verifies that the image copy, change accumulation, and logs are consistent, and it checks that the timestamp in the image copy header record matches the timestamp for the image copy record in RECON.

It selects log records with a timestamp later than the image copy header record. Log and change accumulation input must be in chronological order, (i.e., timestamps may not go backwards). GENJCL.RECOV detects any overlapping logs that require merging as described above and produces a message saying if merge is needed. You simply run change accumulation (or if you have been clever enough to buy RECOVERY PLUS, it does it all for you).

Changing the Clock

This section discusses methods for resetting your OS clock prior to V6. If you are totally on V6, and the RECONs are not going to be used by V4 or V5 ever again, then it is safe to move the clock forwards and backwards while IMS is running. It is *not* safe to do this if your RECONs are being used by V6 and a previous release, or you have to fallback to a previous release—see "Appendix ATHT Problems APAR—PQ03038" on page 193 for more details of how you can make a big mess.

Forward

There are not any special considerations when setting the clock forward.

Backward

Once a year, most locations change the clocks back one hour. This resetting of the OS clocks has serious implications for anyone running IMS subsystems during this period, since database recovery is sensitive to the log record time stamps.

The words “IMS subsystem” appear many times in the following text. IMS subsystems include IMS and DBCTL online systems, IMS DLI/DBB batch jobs, CICS-DL/1 systems, the IMS database utilities (image copy, change accumulation, recovery, unload, reload, prefix update, etc.), and the logging utilities (log recovery and archive). The same considerations apply to change accumulation and the log utilities (log recovery and archive), although they are not strictly IMS subsystems.

Non-DBRC User

A number of options exist for the IMS user not using DBRC. These include the following:

1. Don’t run any IMS subsystems during the duplicate time period (the one hour period between the original 2:00 A.M. and the new 2:00 A.M.). This option is by far the easiest to implement.

2. Bring all IMS subsystems down immediately prior to resetting the clocks, reset the clocks, back up all databases updated in the preceding hour, and restart the subsystems.

3. If no time is available for backing up the databases between the time the subsystems are brought down, the clocks reset, and the subsystems restart, you must remember which logs were created under daylight savings time and which logs were created under standard time.

   Any database recoveries that use a backup created under daylight savings time have to be performed in two steps. The first step recovers the database up to the time change and uses the backup (image copy) and logs created prior to the time change. Once that is done, a second recovery is performed using the logs created after the time change.

All IMS subsystems must be terminated prior to the time change. IMS subsystems generally use the hardware clock (STCK instruction) rather than the OS TIME macro to obtain the time. At subsystem initialization time, IMS computes an adjustment factor to convert the STCK value to a date and time. This adjustment factor is recomputed at midnight. If you reset the OS clock while an IMS subsystem is running, the subsystem does not recognize the time change immediately. However, IMS subsystems started after the time change use the new time. The new subsystems are an hour behind the old subsystems. In addition, if the subsystems running prior to the time change continued execution past midnight (old time), they reset their internal clocks back an hour and cause negative time breaks in the log.

My recommendation—stick to option 1.

DBRC User—Databases Registered

As a DBRC user, you do not have as many options. Almost all of the record types in the RECON data sets have timestamps. Some of these timestamps are provided by the IMS modules that invoke DBRC and some are obtained by the DBRC code (via the TIME macro). DBRC assumes that time never goes backwards and is coded with that basic assumption. Therefore, you must terminate all IMS subsystems prior to resetting the OS clock and wait an hour before running any new IMS subsystems. There are no other options available if you expect DBRC to perform a correct recovery.

DBRC User—Log Control Only

If you are only using DBRC to manage the online logs (i.e., using IMS V1.3 or later, no databases registered), you have the same options as the non-DBRC user with the following restriction if options 2 or 3 are selected.

When the online system is brought down, the user must switch to a clean set of RECONs (i.e., no PRIOLDS, PRISLD, or PRILOG records for the subsystem) and then bring the online system back up with a cold start.

In other words, stick to option 1.

Chapter 16   Testing and Migration

This chapter considers the implications of moving databases from test to production, and how to handle multiple sets of RECONs. It also considers some of the implications of moving from CICS-DL/1 to DBCTL.

Database Names

The rules of DBRC require the database names in RECON be unique. Also, the combination of DBDNAME+DDNAME must be unique. This means you cannot have test and production databases registered in the same RECONs if they have the same names.

This has caused major problems at many CICS-DL/1 sites, where they currently run cloned systems—one CICS-DL/1 region for each application, with the DBD names and ddnames the same, but using different data set names. When they try to combine this all into one DBCTL system, the problem is that the DBD names have to be unique in the IMSGEN (DATABASE macros). If you want to run without DBRC registration, then you could write an exit to point at different data sets depending on keys say, but if you want to gain the benefits of DBRC, then I don’t see an alternative to renaming things.

Most people run with two sets of RECONs—one for test and one for production. They use RACF or something similar to ensure people access the correct set of RECONs.

Test and Production

If you do not have FORCER specified, you get a warning message (database not registered) when running with the wrong database data sets (i.e., do not match what you have registered in RECON). The job will continue to run. You must use other methods, such as RACF, to ensure that you do not use test databases against the production RECONs and vice versa—see my warning in “FORCER” on page 30.

If you have FORCER, the job will abend with an authorization failure if the data set name does not match.

Dynamic Allocation

DBRC uses dynamic allocation for the RECONs if no DD statements are present. If a RECON1 DD statement is present, JCL allocation is used. So to use different sets of RECONs, either generate different sets of dynamic allocate modules and STEPLIB to the appropriate ones or use JCL overrides for test.

When I am debugging RECON problems (oh, what a thrill!), I typically take a copy of the RECONs as a first step. I then use DD statements to point at my copy RECONs and play around with these until I have cracked the problem (nothing worse than entering the wrong command against the production RECONs and then spending an hour undoing it). When I have the job stream all correct, I then asterisk out the DD statements and run against the production RECONs.

Note: One thing that annoyed me intensely when I first started playing with DBRC was that if I entered a string of commands and one of them was wrong, DBRC would stop rather than skipping over it and doing the following ones. Actually, this is very sensible, as it enables you to build things up slowly and in the correct order and not make a total and utter mess of things.

A mixture of dynamic and JCL allocation is not recommended. Imagine the following scenario:

1. Job 1 uses dynamic allocation for all three RECONs.

2. Job 2 has DD statements for RECON1 and RECON2 (forgotten or unaware of RECON3, or copied some test JCL—test regions often run with no SPARE).

3. Job 1 gets error on RECON1, triggers copy to spare, and is now running on RECON2 and RECON3.

4. Job 2 accesses RECON1 and RECON2, finds they don’t match, and stops using RECON1.

5. Job 1 accesses RECON2 and RECON3, finds they don’t match, and stops using RECON3.

So you are rapidly reduced to one RECON. Hopefully, you have coded the parameter NONEW on your RECONs, and no more jobs will start to add to the confusion. For instructions on how to get back to two RECONS and a spare, see “Recovery Following a Discarded RECON” on page 26.

Migration

There is no facility in RECON to merge information from one set of RECONs into another. So when moving databases from test to production, the best thing is to register them, and then image copy them to give DBRC a starting point.

DBCTL

As from IMS V3, the CICS/VS user can access DL/1 databases using the DBCTL address space. Use of this facility means that DL/1 logging, archiving, etc., is performed by DBCTL. The CICS journal only contains CICS information. It does not include any DL/1 database information. See Appendix B, “The Real Rules of Archive” for information on how archive works.

DBCTL users are normally new to IMS and have to suffer some of the things we old IMS lags went through some years ago. The IMS manuals are a good source of information—DBCTL simply assumes that you understand things like IMS logs. Some of the more common problems I come across are:

1. You have to learn about OLDS, SLDS, RLDS, and WADS. Do not forget to set up WADS—they are vital to your performance. Their size in tracks is (1+(OLDSBLKSIZE/2)) x number of OLDS buffers. Put them on high-performance non-volatile storage.

2. You do not have to install a separate DBCTL if you already have an IMS TM control region. The IMS control region contains DBCTL—the manuals hint at this and then tell you no more. Trust me, I know many customers running IMS TM as their "DBCTL server."

3. You have to set up your archiving scheme (i.e., decide on how many SLDS and RLDS you will produce at archive time) and rewrite your recovery procedures to use these as input.

4. You can avoid sharing by using BMPs—recommended—but don't forget checkpoint/restart—BMC Software's APPLICATION RESTART CONTROL can help you implement and control this much more quickly.

5. You can use Fast Path (still the best performing and most available DBMS).

6. You do not have Resource Definition Online—you have to use IMS's rather nasty Online Change feature with all its active, inactive and staging libraries. Alternatively, you could do yourself a favor and have a look at BMC Software's DELTA IMS® for DBCTL product, which allows you to make dynamic changes.

Chapter 17 N-Way Sharing

N-way sharing based on the new Coupling Facility hardware is much easier than the old software-based block-level sharing. It is, however, still inherently complex, and not something to be entered into one wet Tuesday afternoon, because you are feeling bored. Careful planning and a full implementation of DBRC Share Control are prerequisites.

BMC Software will shortly be producing a manual called *IMS DB Backup and Recovery With N-Way Sharing*, which you should read.

General Observations

The major recommendations here are as follows:

- Use an alternative if you can.
  - — IMS TM sites should use BMPs for batch access to online databases.
  - — CICS/VS sites should use the DBCTL feature and look at BMPs.
  - — Sites wishing to share across two CPUs should consider buying a bigger CPU.
- Share between online systems if possible. This provides the following benefits:
  - — easier operations
  - — easier restart
  - — dynamic backout
  - — easier DBRC interface
- Plan carefully, especially in how you are going to distribute the workload. Are you going to have separate systems or cloned systems? Think carefully about naming conventions.

- Test thoroughly.
- Make your operational procedures as simple and as consistent as you can. For example, always close the log, backout, and restart from last checkpoint in batch. Look at BMC Software's BATCH CONTROL FACILITY and APPLICATION RESTART CONTROL products to help you in this area.

Some Areas to Watch

Here are a few suggestions/comments/problems I have heard about from customers looking at/implementing n-way sharing:

- RECON performance

  The RECONs will be shared among all sharing subsystems. Each of these will be doing ENQ/DEQ. It is vital that you tune your RECONs—see Chapter 2, "RECON Setup" for further details—so that they do not become a bottleneck. I expect IBM to offer further solutions to this problem in the future.

- WADS I/O

  I am told that the WADS I/O increases when you are using the IRLM—I don't really know why, but keep a watch on it. WADS are critical data sets, and must perform well.

- PROCOPT=GOT

  There used to be a problem with PROCOPT=GOT flushing the complete buffer pool. This has been fixed via APAR No. PN63464.

  "GO/GON/GOT all cause all buffers to be invalidated. Since only GOT does a retry and buffer invalidation is part of retry it is the only PROCOPT that will impact other on-line users. With PROCOPT=GO the results are an ABENDU0849 and with PROCOPT=GON then a GG status is returned. Only GOT does the retry prior to returning the status GG. In IMS Version 5.1 and IRLM Version 2.1 with (CF) coupling facility structures using PROCOPT GOT causes the entire buffer pool to be flushed after the retry.

  Code was added to module DFSDLR00 that will check to see if VSAM or OSAM coupling facility structure is being used. If the coupling facility structure is being used we do not invalidate the buffers."

- Restarting BMPs on a different IMS

  BMPs expect to be restarted on the same IMSID as they ran. BMC Software's APPLICATION RESTART CONTROL handles this situation.

Recovery

Recovery in n-way sharing brings a whole new set of considerations:

- Finding a timestamp you can recover to is almost impossible (unless you have BMC Software's RECOVERY MANAGER for IMS) as databases tend to be online to at least one subsystem all the time. BMC Software's RECOVERY PLUS recovers to any point in time (PIT) feature solves this problem, and RECOVERY MANAGER for IMS will analyze logs for you to find appropriate timestamps and generate the JCL to recover.

- Databases have to be taken off all sharing subsystems—this means an update to your procedures to use the GLOBAL parameter. RECOVERY MANAGER for IMS automates this procedure, and also checks that the database(s) really is/are offline. (COMMAND COMPLETED only means that your syntax is correct, it does not mean that the command has executed!)

- Logs have to be merged (=change accumulation) prior to recovery unless you have RECOVERY PLUS.

- Any recovery scenario which involves a subset of a log is *impossible* with the standard IMS utilities. The recovery utility is capable of rejecting unnecessary log records, but the change accumulation always includes complete logs. This is described in APAR No. II09694, and here is one of the scenarios, which is *not* supported by the standard utilities:

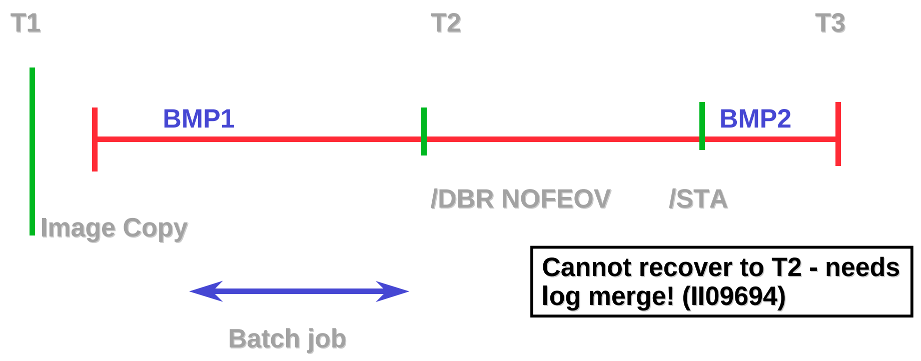


Figure 58 Impossible Recovery with Standard Utilities

In this scenario, a /DBR NOFEOV has been performed on the online system, e.g. to establish a recovery point. The database is being updated by online—BMP1—and a batch job prior to the /DBR. If you try to recover later on to the /DBR, you will find it is impossible, because you need to run change accumulation to merge the logs, but change accumulation includes the log records after the /STA. GENJCL.RECOV recognizes this and wants to select the logs instead, but it realizes they need merging so it abends with CC=12.

The resolution to the APAR is to amend the *IMS/ESA V5 Utilities Reference* manual to say:

"A valid timestamp for a partial recovery is any point at which there are no allocations of the DBDS or area and there is not a merge of logs needed that cannot be resolved by running the accumulation utility."

In other words—avoid the problem!

Fortunately, these problems do not occur with RECOVERY PLUS, as this product will sort/merge and portions of logs as required and reject any unnecessary log records.

Disaster recovery procedures have to be rewritten—see Chapter 18, "Offsite Considerations."

Chapter 18 Offsite Considerations

Note: When I first wrote this chapter in the original version of the manual, it was all theoretical (but I did point that out!). The good news is that all of this has been tested and implemented by various customers round the world. My thanks to them for trying it out and letting me know what I had got right/wrong and how to improve the procedures.

I am glad to say that more and more customers have now implemented (and tested) disaster recovery procedures. While I hope that you will never need this chapter, experience shows that disasters do happen, and the companies that do not have a plan in place suffer severe financial loss, due to the lack of a system for an extended period of time.

Interestingly, a significant number of the disasters I get involved in never actually get offsite. Many recoveries nowadays tend to be for logical or human reasons rather than physical (hardware is much more reliable, but not 100%), so your procedures have to cater for this as well.

The Fundamental Questions

Before planning any offsite procedures, you must ask yourself some fundamental questions.

- How far am I prepared to go back, (i.e., how many hours of work am I prepared to lose?)
- How long do I have to recover at the backup site?
- What impact can I tolerate at the prime site to enable this (e.g. can I take databases offline to copy them?)
- How much am I prepared to invest in offsite?

Answers to the above vary from, “I’ll go back to last night at 6 o’clock and allow 24 hours to recover” to “As fast as possible, with *no* loss of data.”

The first is eminently achievable. The second is an area many customers hoped to solve via duplexed DASD. This is not without its own problems though:

- Do I run synchronously?

  If so, what are the performance implications, especially on data sets like the WADS?

- Is it truly synchronous?

  Depending on the hardware, in some cases it is semi-synchronous, or only synchronous within one control unit. Talk to your vendor and understand what is really happening.

- Is asynchronous any use?

  Are the databases in sync with the log/one another/another DBMS? The fundamental source of correct data in any IMS system is the log. You can rebuild *everything* from the log and restart/backout etc. If you want to duplex any data sets, then the logs (and the RECONs) should be the highest priority.

  One golden rule of DR is that the data offsite—e.g., logs/image copies—must match the RECONs offsite, or in other words whenever and however you take data offsite, take the RECONs at the same time.

- What happens if I lose the connection?

  Does all I/O stop? Does it ignore? Do I get into a state of catch-up where I don’t know what is offsite? Does the data arrive offsite in the correct order?

In all cases DBRC and the RECONs add a new dimension to the planning for offsite. In all your contingency planning, you must consider data set names, catalogs, etc., and how you are going to synchronize these between sites.

This chapter does not consider MSDBs, as there is no DBRC support for them. Take copies of the checkpoint data sets offsite or convert them to VSO DEDBs. It also does not consider GSAM yet, but I’m working on a solution for this.

Basic Principles

To start with, I shall consider two common options:

- recovery to last night's image copies
- recovery to the end of the last good log

The initial discussions do not include n-way sharing or multiple DBMSs. This is considered at the end of the section.

Several of these scenarios involve closing log records in RECON. The offsite RECONs are going to indicate the fact that IMS is running and the log is OPEN (STOPTIME of zeroes). DBRC will not let you generate recovery JCL if it thinks the databases are currently allocated to a running subsystem—ALLOC points at an OPEN PRILOG, SUBSYS not flagged as ABNORMAL TERM. Various techniques for curing these problems are given below and some of them involve closing the log, i.e., inserting a STOPTIME. Or you can use BMC Software's RECOVERY MANAGER for IMS RECON Cleanup utility.

Version 5 Changes

As from V5, there are two new parameters on the log commands—FIRSTREC and LASTREC. I don't know why they were introduced—my guess is something to do with RSR and tracking which log records have been shipped offsite. (I met the man from the IBM DBRC Change Team the other week, and he confirmed that is what FIRSTREC and LASTREC are used for—they are not used in a non-RSR environment.) The significance here is that they are required on certain commands.

First of all a definition—for instance, on the OLDS commands:

FIRSTREC(number)

is an optional parameter you use to specify the log record sequence number of the first log record of the OLDS. For the first OLDS of the PRILOG, it corresponds to the first log record written during initialization of the IMS subsystem.

FIRSTREC is required for OLDS Open and Switch commands. It specifies the first log record sequence number on the OLDS being opened. It is invalid for a Close command.

The log record sequence number can be entered as a hexadecimal number e.g. FIRSTREC(X'10B9C'), or as a decimal number, e.g., FIRSTREC(68508).

LASTREC(number)

is an optional parameter you use to specify the log record sequence number of the last log record of the OLDS. LASTREC is required for the OLDS close command. It is optional for the switch command; if it is omitted, the FIRSTREC value of the next OLDS minus one is recorded for the OLDS being closed. It is invalid for an OPEN command.

I have played with these and in my testing I simply looked at the FIRSTREC value in RECON (which you can extract via %SELECT OLDS—%OLDFRID gives you FIRSTREC and %OLDLRID gives you LASTREC), added one to it and put that in as LASTREC.

Test Results

Test results for FIRSTREC and LASTREC follow.

```
TEST1: Specified:   OPEN:  FIRSTREC(1)
                  SWITCH:  LASTREC(4999)
                           FIRSTREC(5000)
                   CLOSE:  LASTREC(9999)

Results             LOG1:  FIRSTREC(X'0001')
                           LASTREC(X'1387')
                    LOG2:  FIRSTREC(X'1388')
                           LASTREC(X'270F')
```

Figure 59 Base Case Works!

```
TEST2: Specified:   OPEN:  FIRSTREC(1)
                  SWITCH:  LASTREC(4999)
                           FIRSTREC - omitted
                   CLOSE:  LASTREC(9999)

Results: OPEN executed okay
         SWITCH received:  "Required FIRSTREC parameter
                            not specified"
```

Figure 60 SWITCH without FIRSTREC Specified Doesn't Execute

```
TEST3: Specified:   OPEN:  FIRSTREC(1)
                  SWITCH:  LASTREC - omitted
                           FIRSTREC(5000)
                   CLOSE:  LASTREC(9999)

Results: Same as TEST1, i.e., LASTREC given value of
         FIRSTREC - 1
```

Figure 61 SWITCH without LASTREC Specified Assigns LASTREC = FIRSTREC - 1

```
TEST4: Specified:   OPEN:  FIRSTREC(1)
                  SWITCH:  LASTREC(4999)
                           FIRSTREC(5000)
                   CLOSE:  LASTREC - omitted

Results:  OPEN and SWITCH executed okay
          CLOSE received:  "Required LASTREC parameter
          not specified"
```

Figure 62 CLOSE without LASTREC Doesn't Execute

Recovery to Last Night’s Image Copies

In this scenario, dual image copies are taken and one copy is shipped offsite. In the event of a disaster, complete the following steps:

Step 1 Recover the databases from the image copies.

If you have a secondary IC, then you must flag the primary as INVALID, or simply use the SIC parameter in BMC Software’s RECOVERY PLUS. See “Timestamp Recovery” on page 124 for details of automating recovery back to the last image copy.

Fast Path DEDBs must be flagged as recovery needed to enable recoveries.

The recoveries can be done outside DBRC using standard JCL decks calling in the latest generation of image copy. However, I think you should use DBRC (why go to all that effort of implementing DBRC and then not use it?) so you should ship a copy of the RECONs with the image copies, restore the RECONs and use the procedures detailed above. Or use BMC Software’s RECOVERY MANAGER for IMS Recover to Last Batch IC option.

One of the golden rules of recovery with DBRC is that the data and the RECONs should be taken offsite at the same point in time. If the RECONs don't match the data, then they are basically useless.

If your BACKUP of RECON shows SUBSYSTEMS running, they will have to be cleaned up. Instructions for this cleanup follow.

Step 2 Start online and batch work as appropriate.

The databases are now consistent, so work can start.

Recovery to the End of the Last Good Log

In this scenario, image copies and logs (and maybe change accumulations) are shipped offsite.

Note: You cannot run backout from a change accumulation file or an RLDS, because they do not contain any beginning and end of transaction records, checkpoints, etc.

If you have a disaster and recover the databases to the end of the last good log, they will be consistent if it is a batch log, but will be inconsistent if it is an online log (unless the last log is the end of the on-line session), unless you have RECOVERY PLUS and use the recover to *any* point in time option—this is covered later in this chapter.

If it is a batch log, proceed as above. If it is an online log in the middle of online IMS, the actions depend on what level of DBRC you are using. Some manuals recommend taking a previous backup of RECON and rebuilding the PRILOG/PRISLD, etc., records via command (actually, you cannot build all the information you require, e.g., DSSN, USID, which means that RECONs built in this way can lead to corruption of the databases). This is an extremely boring, time-consuming, tedious, and error-prone procedure and is definitely not recommended as it does not work. The person who wrote this procedure obviously never tried it!

Log Control

The recoveries have to be performed outside DBRC control. DBRC has no knowledge of image copies, etc. When the databases have been recovered, you can either:

- Run batch backout for the transactions that were in flight at the end of the last log (DFSULTR0 with PSBLIST=YES will give you a list) and cold start the system with a fresh set of RECONs.

OR

- Take a copy of the RECONs with you with every log and use these to emergency restart. You want to force IMS to start from the SLDS (the OLDS are not available) so you have to delete the PRIOLD and SECOLD record in RECON. IMS will dynamically allocate the last SLDS if it can't use the OLDS and restart from it. You must have a dynamic allocate macro set up for the SLDS (using DFSMDA) to make this work.

  The commands to delete the OLDS records are:

  — NOTIFY.PRILOG OLDS(..) STARTIME(..) RUNTIME(..) SSID(..) to close the last open OLDS. As from V5, also code the LASTREC parameter

  — CHANGE.PRILOG OLDS(..) SSID(..) ARCHIVED for any unarchived OLDS

  — DELETE.LOG OLDS(..) SSID(..) for all but the last one

  — DELETE.LOG OLDS(..) SSID(..) LASTCLOS for the last OLDS you closed above

  OR

- Take a fresh set of RECONs and NOTIFY the last two SLDS and use /ERE.

  Emergency restart always asks for the primary SLDS even if it is marked in ERROR in RECON—this has been APAR'd, but the response is that it is working as designed. If you only have your secondary SLDS you will have to change the PRISLD record to point at it using CHANGE.PRILOG.

Recovery Control

If you restore an old set of RECONs, the information will be inaccurate and basically useless (see "Losing All Three RECONs" on page 26 for additional information), so don't! You should take a copy of the RECONs offsite with each log.

You need to restore the databases. DBRC will not allow you to do this if the databases are ALLOCated on an OPEN log (unless you have RECOVERY PLUS—see later). The options here are either:

- Close the logs.

In my testing, DFSULTR0 does not put a STOPTIME in the RECONs. You have a PRILOG/PRISLD record with a STARTIME of T1, and a STOPTIME of zeroes. Within the record are multiple records, each with valid START and STOP times:

```
e.g. PRISLD START=T1 STOP=00.000 00:00:00.0
     entry1 START=T1 STOP=T2
     entry2 START=T2 STOP=T3
     entry3 START=T3 STOP=T4
```

Figure 63 Offsite PRISLD/PRILOG Record

DFSULTR0 will happily close entry 3, but it does not put anything in the STOPtime at the top of the record—this stays at zeroes because DBRC is expecting you to run another archive.

There are two ways to close these records:

1. Add another entry (for a dummy empty log). The commands to do this are:

```
NOTIFY.PRILOG SLDS SSID(..) STARTIME(T1) DSN(....) VOLSER(...)

NOTIFY.PRILOG SLDS SSID(..) STARTIME(T1) RUNTIME(T5)
```

The dummy log should be a null data set in VB format with no records in it. As from V5 you will need FIRSTREC on the first command and LASTREC on the second.

OR

2. Run archive. If IMS had not “gone bang,” the next thing that would have happened is that the current OLDS would have filled up and been archived. So the easiest thing to do is to simulate this. First, you look in RECON to see which is the current ACTIVE OLDS (you can use %SELECT OLDS INUSE to find which one). It will be flagged in RECON as ACTIVE with a STOPTIME of zeroes, and the associated PRILOG/PRISLD etc. records will also have STOPTIMEs of zeroes:

```
PRILOG
 START   = 93.341   23:47:57.3*  STOP    = 00.000   00:00:00.0
 SSID=DVT1         #DSN=1        IMS

  DSN=CSG.IMSB.RLDSP.D93341.T2347573.V00 UNIT=SYSALLDA
  START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
  FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
                                   WORKB1    93.342   06:05:11.0
------------------------------------------------------------
PRISLD
 START   = 93.341   23:47:57.3*  STOP    = 00.000   00:00:00.0
 SSID=DVT1         #DSN=1

  DSN=CSG.IMSB.SLDSP.D93341.T2347573.V00 UNIT=SYSALLDA
  START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
  FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
                                   WORKB1    93.342   06:05:11.0
-----------------------------------------------------------
PRIOLD
 SSID=DVT1            # DD ENTRIES=2

 DDNAME=DFSOLP00   DSN=CSG.IMSB.OLP00
  START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
  STATUS=ARC COMPLT                               FEOV=YES   AVAIL
  PRILOG TIME=93.341   23:47:57.3      ARCHIVE JOB NAME=LOGDVT1

 DDNAME=DFSOLP01   DSN=CSG.IMSB.OLP01
  START   = 93.342   06:05:11.0   STOP    = 00.000   00:00:00.0
  STATUS=ACTIVE                                   FEOV=NO    AVAIL
  PRILOG TIME=93.341   23:47:57.3
```

Figure 64 Offsite RECONs with OPEN Logs

To archive the ACTIVE OLDS—DFSOLP01, it must be closed and ARCHIVE NEEDED. I used the following command:

```
NOTIFY.PRILOG OLDS(DFSOLP01) SSID(DVT1) STARTIME(933420605110) -
RUNTIME(933420605111)
```

This command closes the OLDS, which also flags it as ARCHIVE NEEDED. I simply set the RUNTIME to 0.1 second later than the STARTIME, but any time will do—for instance in my *GENJCL.USER Examples and Description* manual I show an example with GENJCL.USER and some skeletal JCL like this:

```
%SELECT OLDS(DVT1,INUSE)
%DELETE (%OLDSTYP EQ 'S')
NOTIFY.PRILOG OLDS(%OLDSDDN) STARTIME(%OLDOTIM) -
RUNTIME(%DATE%TIME%CNTR)
%ENDDEL
%ENDSEL
```

Figure 65 Using GENJCL.USER to Close the Last OLDS

As from V5, you would have to put in LASTREC as well.

The OLDS is now flagged in RECON as closed and ARCHIVE NEEDED, so I walked up to ISPF and allocated an empty OLDS1 and archived it. The Archive utility found no records for the SLDS or RLDS, but it still runs (and I’ve tested this on V6!), and puts STOPtimes in everywhere for you.

```
PRILOG
 START   = 93.341   23:47:57.3*  STOP    = 93.342   06:05:11.1

 DSN=CSG.IMSB.RLDSP.D93341.T2347573.V00 UNIT=SYSALLDA
 START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
 FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
                                  WORKB1    93.342   06:05:11.0

 DSN=PWA.IMSB.SLDSP.D93342.T0605110.V00 UNIT=SYSALLDA
 START   = 93.342   06:05:11.0   STOP    = 93.342   06:05:11.1
 FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
                                  WORKB1    93.342   06:05:11.1
----------------------------------------------------------------------
PRISLD
 START   = 93.341   23:47:57.3*  STOP    = 93.342   06:05:11.1
 SSID=DVT1         #DSN=2

 DSN=CSG.IMSB.SLDSP.D93341.T2347573.V00 UNIT=SYSALLDA
 START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
 FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
                                  WORKB1    93.342   06:05:11.0

DSN=PWA.IMSB.SLDSP.D93342.T0605110.V00 UNIT=SYSALLDA
START   = 93.342   06:05:11.0   STOP    = 93.342   06:05:11.1
FILE SEQ=0001    #VOLUMES=0001  -VOLSER-        -STOPTIME-
WORKB1    93.342   06:05:11.1
----------------------------------------------------------------------
PRIOLD
 SSID=DVT1            # DD ENTRIES=2

 DDNAME=DFSOLP00   DSN=CSG.IMSB.OLP00
  START   = 93.341   23:47:57.3   STOP    = 93.342   06:05:11.0
  STATUS=ARC COMPLT                                FEOV=YES   AVAIL
  PRILOG TIME=93.341   23:47:57.3      ARCHIVE JOB NAME=LOGDVT1

  DDNAME=DFSOLP01   DSN=CSG.IMSB.OLP01
  START   = 93.342   06:05:11.0   STOP    = 93.342   06:05:11.1
  STATUS=ARC COMPLT                                FEOV=NO    AVAIL
  PRILOG TIME=93.341   23:47:57.3      ARCHIVE JOB NAME=PWAARCH
```

Figure 66 Offsite RECON with “Closed” Logs

- Pretend the databases were deallocated just before the end of the last log using NOTIFY.ALLOC ... DEALTIME(log stop - 1). This is documented in my *GENJCL.USER Examples and Description* manual, but as I say in there as well, I find the dummy archive a much easier method.

If the logs are closed or the databases are deallocated, you can use DBRC; if not, you have to run the recoveries outside DBRC.

If you are using secondary image copies, you must flag the primaries as INVALID. If you are using secondary logs, you must flag the primaries as ERROR. BMC Software's RECOVERY PLUS has two parameters—SIC and SLOG—which automatically select secondary data sets without having to flag the primaries as invalid. Fast Path DEDBs must be flagged as recovery needed.

You must also delete the PRIOLD and SECOLD records from RECON as the OLDS are not available for restart. Or use BMC Software's RECOVERY MANAGER for IMS RECON Cleanup utility.

You now need to run backout and cold start for DL/I and log close, forward recovery for Fast Path or use the /ERE technique as in log control above.

If you use RECOVERY PLUS *any* point-in-time recovery (PIT) to recover to the end of the last log, the databases will be consistent and *no further cleanup* is necessary. Only complete transactions will be applied, both for full-function and Fast Path.

NONRECOV

If you are using NONRECOVerable databases, then you must remember that an archived OLDS (i.e., a SLDS) does not contain any *before* images for NONRECOVerable databases, and hence backout is impossible for these. The correct actions would be:

- Recover the prime databases.
- Cleanup via batch backout or /ERE.
- Restore the NONRECOV databases using, for instance, BMC Software's SECONDARY INDEX UTILITY/EP.

With RECOVERY PLUS PIT, you simply recover everything to the end of the last log and all the databases will be consistent. Lastly, when using the RECOVERY PLUS PIT feature for a NONRECOV index, you must use either the BLDINDEX(Y) or SIUDSN keyword to ensure the integrity of the index. See "RECOVERY PLUS and SECONDARY INDEX UTILITY/EP" on page 142.

Share Control

Everything discussed above applies here as well, but Share Control introduces the extra complication of SUBSYS records and authorizations.

Use LIST.SUBSYS to find the active subsystems in RECON and issue the CHANGE.SUBSYS ... ABNORMAL commands for them. This will enable recovery to run.

The various types of SUBSYS that might be in your RECONs are:

- utilities, batch read jobs

  CHANGE.SUBSYS ... ABNORMAL will remove these

- batch updates

  CHANGE.SUBSYS ... ABNORMAL, CHANGE.SUBSYS ... STARTRCV, and CHANGE.SUBSYS ... ENDRECOV will remove these.

  Of course, you also have to clean up the batch job by closing the log (if you have it) and batch backout, or restore to the end of the last good log offsite.

- online systems

  CHANGE.SUBSYS ABNORMAL followed by an unplanned cold start, or an /ERE from the SLDS.

Change Accumulation

If you are doing regular CAs at your prime site, your RECONs will show this. So take a copy of the CA file offsite (using IEBGENER or BMC Software's CHANGE ACCUMULATION PLUS) and update the offsite RECONs to point at it with a CHANGE.CA command. If you don't do this, you will have to flag the CA records as INVALID to force DBRC to use the logs.

Warning! Remember that batch backout will not run from a change accumulation data set or a RLDS—you need the SLDS offsite, if you are going to run backout. RECOVERY PLUS PIT also needs SLDS in the current release (April 1998).

Shadow Databases

The major difficulty in maintaining shadow databases with DBRC is that intermediate logs are not valid recovery points. Hence, the recoveries have to be run outside DBRC or you have to close the log records in RECON or DEALLOCate the databases prior to running the recoveries.

One customer uses the following commands as each set of logs/RECONs comes over:

- NOTIFY.ALLOC ..... DEALTIME(log stoptime -1)
- NOTIFY.UIC .... RUNTIME(log startime) and
  NOTIFY.RECOV .... RCVTIME(log startime)

  These commands tell DBRC that the DBDS was timestamp recovered to the beginning of this log = end of last one

- CHANGE.SUBSYS .... ABNORMAL for IMS
- CHANGE.DBDS .... RECOV for the DEDBs
- GENJCL.RECOV ..... USEAREA/USEDBDS

  This command applies the last log on top of the shadow database.

Warning! Fast Path users must include the last log and the previous one each time. The recovery utility only applies complete transactions for Fast Path DEDBs, so there will probably be an incomplete chain on the previous log, which finishes on this one—hence, you have to put the previous one in each time to pick up the first part of the chain.

N-Way Sharing/Multi-DBMS

The problem with more than one DBMS (e.g., 2 x IMS or IMS and DB2) is that the logs do not switch at the same time and, hence, it is normally impossible to find a point that you can recover to. These problems do not apply only to those with transactions that update IMS and DB2—the same problem exists if you read IMS and update DB2. Recovery of one DBMS can be accomplished as detailed above—it is achieving consistency with two or more DBMSs that brings the problems.

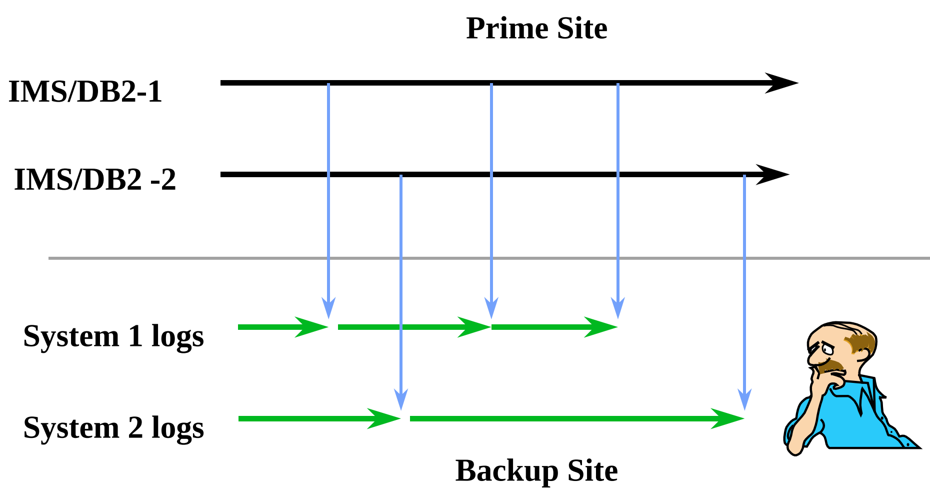


Figure 67 Trying to Find a Recovery Point in a Multi-DBMS Environment

A common approach used here is to shut down all the subsystems at regular intervals and ship this recoverable position offsite. BMC Software's RECOVERY MANAGER for IMS and RECOVERY MANAGER for DB2* will do this for you across one or more IMS and/or one or more DB2 systems.

However, you may not want to have the interruption that this gives, so the RECOVERY MANAGER products offer a second option, which is to synchronize the log switches across all the subsystems.

Some customers are looking at hardware to solve this problem, but see the comments above.

RSR and RRDF

IBM announced RSR in IMS V5 and ENET provide the Remote Recovery Data Facility (RRDF) program, which allows IMS log data and pertinent DBRC information to be transported in real-time mode to a remote site, enabling the remote site to recover IMS databases if an extended primary outage occurs.

With RSR and RRDF you can run in one of two modes—shadow logs or shadow databases. RSR supports n-way sharing, but does not support DB2 or VSAM and is covered in the BMC Software manual *RSR in Practice*.

RRDF supports IMS, DB2, and VSAM. The basic principle is that it creates shadow RECONs and shadow SLDS at the second site. The implications are the same as recovering to the end of the last log. You would then run /ERE to perform the dynamic backout of the in-flights as you have the SLDS available.

RECOVERY PLUS

RECOVERY PLUS will automatically select secondary image copies and/or logs for you via the parameters SIC and SLOG. One of the standard functions of RECOVERY PLUS is to maintain shadow databases at a contingency site.

RECOVERY PLUS recovery to *any* point in time (PIT) means that you do not have to shut down systems at all. You simply take the latest logs offsite and recover to the latest common transaction boundary. The RECOVERY MANAGER products automate this process for you and also handle coordinated consistent recovery between multiple DBMSs.

Summary

This is an area requiring careful planning and extensive testing. Try to automate as much of the process as possible using the GENJCL.USER and %SELECT, %DELETE facilities.

If you don’t want to take the RECONs with you, you have to recover the databases, run the backouts, take a fresh set of RECONs, and then image copy (or use IMAGE COPY PLUS) *everything* to give DBRC a starting point, and then cold start.

BMC Software’s recovery products make the whole process much easier, much quicker and enable coordinated consistent recoveries between DBMSs.

Appendix A THT Problems APAR—PQ03038

Note: To look at closed APARs, I simply logon to IBM's website, which is where I got this.

PQ03038: 6.1—Pre-6.1 Coexistence Problems:

- THT built with incorrect range.
- RECON local times ambiguous after clock change.

For pre-6.1 DBRC to function properly with a 6.1 RECON, the downgrade and upgrade of any RECON timestamp must be symmetrical—that is, a UTC timestamp downgraded to local time must yield the original UTC time if it is subsequently upgraded. Two conditions can cause a non-symmetrical downgrade/upgrade:

- A THT entry has an incorrect range.
- Timestamps exist in the RECON whose local times are ambiguous.

The time history table (THT) in the 6.1 RECON contains a history of time-zone changes. This is basically a record of when the system time was changed to and from Daylight Savings Time. The THT is used (1) in the RECON Upgrade process (PGM=DSPURU00) and (2) in the RECON Coexistence environment to enable pre 6.1 DBRC code, with the proper RECON Migration and Coexistence support applied, to convert pre-6.1 timestamps to the UTC format.

You need the THT to upgrade a 4.1 or 5.1 RECON to 6.1. If you share a 6.1 RECON with a 5.1 and 6.1 system, then you need the THT—this is coexistence, and you most likely upgraded the 5.1 RECON to a 6.1 RECON by following the RECON upgrade procedure. To coexist, you need the RECON Migration and Co-existence APAR (PN89491 or PN89490) applied. With it, DBRC downgrades timestamps it reads from the 6.1 RECON to the form for processing and then upgrades the 7-byte form to UTC format using the THT. This timestamp processing affects all DBRC processing (including database recovery).

Note: The execution of the 6.1 INIT.RECON command will create a THT record.

You create the THT with the INIT.RECON command in 6.1. You change it with the CHANGE.RECON THT (or REPTHT) command in 6.1. See the 6.1 *DBRC Guide and Reference* for more details.You change the THT whenever you change the system clock and you tell DBRC by issuing the CHANGE.RECON THT command. If you are not coexisting with pre-6.1 DBRCs and never intend to—i.e., you do not need a fallback capability—then you do not need to worry about the THT.

Note: With PQ03038 applied, if in NOCOEX mode, the THT table is not listed when LIST.RECON STATUS or LIST.RECON is done.

Users Affected

All IMS V6 users who coexist with V4 or V5.

Problem Description

In a coexistence environment, there are two conditions that can cause a non-symmetrical downgrade/upgrade of a timestamp. One condition occurs when a THT entry has an incorrect range. The other condition occurs when timestamps exist whose local time values are ambiguous.

What Is DBRC Coexistence?

DBRC coexistence is the accessing of a V6 RECON data set by a pre-V6 IMS (hereafter referred to as V5), including:

- access by both V6 and V5 IMS
- fallback to V5 processing but retaining the V6 RECON

The Role of the THT in Coexistence

When a V5 IMS reads a V6 RECON, it is a virtual V5 RECON; that is, the RECON I/O interface converts all records retrieved to V5 records. As part of this conversion, all timestamps are downgraded—converted to the old, 7-byte local-time format, NOT UTC. In this process, the offset from UTC, after being used to obtain the local time, is lost. When a V5 IMS updates or writes a RECON record, it presents a V5 record to the RECON I/O interface, which converts it to a V6 record. In this conversion, timestamps added or changed by the caller are upgraded—converted from the 7-byte local-time format to the V6 UTC format.

The offset required for this conversion cannot be supplied by the V5 code, so it is obtained from the THT. Even when reading a V6 RECON, many retrieval calls, such as NEXT, involve the passing of record keys to the RECON I/O interface. These keys are in V5 format and may contain timestamps. These timestamps are upgraded in the same way as described above before the retrieval operation is done; thus, even reading the RECON requires the use of the THT.

Possible Problems With This Process

The THT, given an input local time, supplies the correct offset to convert it to UTC. The coexistence process assumes that any timestamp obtained from a V6 RECON and downgraded to V5 can be subsequently upgraded to the original UTC time. In order for this process to work correctly, there are two requirements.

- The entries in the THT must accurately represent the time when each local-time change (normally from standard to daylight time or vice versa) occurred relative to timestamps in the RECON data set.

- The RECON must contain no UTC timestamps from different time periods which, when downgraded, fall into the same local-time period. For example, after a local-time setback of one hour, UTC time 09:02 with an offset of -7, and the later 10:01 with an offset of -8, downgrade to local times 02:02 and 02:01 respectively. Not only does this present invalid data to V5 code, it is impossible subsequently to upgrade both timestamps correctly.

Violation of these conditions can occur just as well from V6 processing as from V5. However, the adverse effects, such as endless loops, are felt only by V5.

Avoiding V5 Coexistence Problems

Until such time as you are sure that you do not and will not require any V5 processing of your V6 RECON, you can assure that the above requirements are met by:

- Shutting down any V5 or V6 processing which could update the RECON before changing the local time, and allowing no updates (other than via DBRC commands) by either V5 or V6 before you have entered the CHANGE.RECON THT command.

- After setting the local clock backward, allow no V5 or V6 updates for a time equal to the amount of the setback (usually one hour).

Note: You can quiesce a V6 online IMS such that it does no RECON access, rather than shutting it down.

Enforcement by DBRC

You declare to DBRC when you no longer have any need for V5 processing of your V6 RECON. DBRC enforces the above restrictions before that time.

How V6 Timestamps Are Downgraded To Pre-V6 Form & Vice Versa

This process is done by code in V4 and V5 when reading from and writing to a V6 RECON. (In RSR, it is done by a V6 tracker tracking a V5 active site.)

Note that for conciseness, in all the following diagrams timestamps are shown with only *hhmm* values,   and sometimes offsets from UTC are shown with only the *h* value—for example, *-7*.

```
--V5--------Downgrade----------------V6----
+Local+    UTC +offset        +-UTC--Offs-+
|01:10|<---------------------|09:10|-8:00|
+-----+                      +-----------+
```

The offset in the V6 timestamp is used to convert it to local time in V5 form. The offset is lost in the process, because there is no room for it in the 7-byte V5 format.

```
Upgrade--offset obtained from THT
--V5------------THT-----------------V6------
          +-UTC---Offs-Local+
+Local+   |00:00 -7:00 00:00|    +-UTC--Offs-+
|01:10|-->|09:00 -8:00 01:00|--->|09:10|-8:00|
+-----|   +-----------------+    +-----------+
           Local -offset
```

Each THT entry indicates the time range for a particular offset. The time range of each entry ends where the next one starts, except for the last one, which continues indefinitely. When a timestamp is upgraded, the offset is obtained from the THT entry whose local time range includes the time to be upgraded.

Possible Problems

For pre-V6 IMS to function properly with a V6 RECON, downgrade and upgrade of any timestamp must be symmetrical; that is, a UTC timestamp downgraded to local must yield the original UTC time if it is subsequently upgraded.

There are two conditions that can cause non-symmetrical downgrade/upgrade.

- A THT entry has an incorrect range. In the following diagram, the local clock was set backward at 09:00 UTC, and a new timestamp was created at 09:10 with the new offset of -08:00, but the new entry for offset -08:00 begins at 09:15 (01:15 local).

```
  V5                                 V6
+Local+                          +-UTC--Offs-+
|01:10|<-------------------------|09:10|-8:00|
+-----+                          +-----------+
  V5            THT                  V6
+Local+   +-UTC---Offs-Local+    +-UTC--Offs-+
|01:10|-->|00:00 -7:00 00:00|--->|08:10|-7:00|
+-----+   |09:15 -8:00 01:15|    +-----------+
          +-----------------+
```

An incorrect THT range is also a problem if the clock was set forward. In the following diagram, the local clock was set forward at 09:00 UTC, and a new timestamp was created at 09:10 with the new offset of -07:00, but the new entry for offset -07:00 begins at 09:15 (02:15 local).

```
  V5                                   V6
+Local+                            +-UTC--Offs-+
|02:10|<---------------------------|09:10|-7:00|
+-----+                            +-----------+
  V5              THT                  V6
+Local+   +-UTC---Offs-Local+    +-UTC--Offs-+
|02:10|-->|00:00 -8:00 00:00|--->|10:10|-8:00|
+-----+   |09:15 -7:00 02:15|    +-----------+
          +-----------------+
```

- Timestamps exist whose local times are ambiguous. This problem can only occur when the local clock is set backward In the following diagram the clock was set backward one hour at 09:00 UTC. There is no way to construct a THT that can distinguish one 01:10 local time from the other; every UTC time between 09:00 and 10:00 will be ambiguous when converted to local time without offset.

```
  V5                                 V6
+Local+                            +-UTC--Offs-+
|01:10|<---------------------------|08:10|-7:00|
+-----+                            +-----------+
+Local+                            +-UTC--Offs-+
|01:10|<---------------------------|09:10|-8:00|
+-----+                            +-----------+
```

The Solution

The solution to both these problems is to require that V6 IMS follow the same rules as pre-V6 with regard to local-time changes as long as coexistence with pre-V6 is enabled. These rules will be enforced by message DSP0345 and abend 2475, just as in pre-V6. Obviously, these rules cancel out a principal feature of V6, so we need to stop enforcing them when coexistence is no longer needed. We can't predict the future, so we can't know when coexistence has finally ceased unless the user tells us.

INIT.RECON NOCOEX will tell us that pre-V6 will never access the new RECON. Otherwise, coexistence is enabled.

CHANGE.RECON NOCOEX will disable coexistence. With coexistence disabled, the user does not need to update the THT when the local time is changed, and V6 does not have to follow the shutdown rules.

CHANGE.RECON COEX will allow the user who has made the wrong move to correct it. There would be no harm as long as the local time hasn't changed since he disabled coexistence. Otherwise, serious malfunctions could occur in his pre-V6 systems. We will make no attempt to protect him from this.

Other Related Changes

The RECON-last-accessed time, up to now set only by a coexisting pre-V6 system, will be set also by each V6 access that updates the RECON, and by the upgrade process. Upgrade will set it to the highest timestamp in any RECON record that is not higher than the time of upgrade.

IMS/ESA V6 Messages and Codes

DSP0307I INTERNAL DBRC ERROR. CODE errcode-diagcode

EC0255-003 DSPURI10 detected that the key size of the input RECONs do not all match.

EC0255-004 DSPURI10 detected that the key size of the input RECON is not 32 bytes.

DSP0345I ACCESS TO RECON DENIED—THT hhmm yyyyddd hhmmsst

Explanation: This message may be issued when IMS attempts to update the RECON after the TOD clock has been changed, or the local clock has been changed and coexistence with pre-61 is enabled. It indicates one of two conditions:

1. The local clock was changed but a new THT entry has not been created. hhmm is the offset from UTC in the current THT entry.

2. The local clock was set back and a new THT entry created, but not enough time has elapsed since the most recent RECON update. IMS cannot update the RECON before *yyyyddd hhmmsst* local time.

System Action: Processing is terminated with ABEND 2475 (with no dump).

User Response: Depends on the condition:

Condition 1: Enter the CHANGE.RECON THT command, then restart the IMS job.

Condition 2: Wait until the indicated time before starting the IMS job.

Or, for condition 2, issue the CHANGE.RECON NOCOEX command to disable coexistence with pre-V6.

IMS/ESA V6 DBRC Guide and Reference

For the INIT.RECON command, two optional keywords are added:

- **COEX**

  If neither COEX nor NOCOEX are specified, COEX is the default. It enables coexistence of pre-V6 with the V6 RECON, and imposes the same processing limitations on V6 as on pre-V6 when the local clock is changed, as for Daylight Saving Time.

- **NOCOEX**

  Use this keyword if you do not intend ever to run any pre-V6 jobs against the RECON (this includes fallback to pre-V6 after encountering a problem with V6). It disables coexistence so that there are no limitations on V6 processing when the local clock is changed. You should be sure of the correctness of this decision—it is possible to reverse it, but see the warning under the COEX parameter of the CHANGE.RECON command.

For the CHANGE.RECON command, two optional keywords are added:

- **NOCOEX**

  Use this keyword if you no longer intend to run any pre-V6 jobs against the RECON. It disables coexistence, removing limitations on V6 processing when the local clock is changed, as for Daylight Saving Time. You should be sure of the correctness of this decision—it is possible to reverse it, but see the warning under the COEX parameter.

- **COEX**

  This keyword allows you to reverse the effect of NOCOEX and once again enable coexistence with pre-V6. Use this keyword with extreme caution; if the local clock has been changed since coexistence was disabled, or if you had used the NOCOEX option to recover from an abend U2475 in V6 (not recommended), endless loops and other, unpredictable, malfunctions in your pre-V6 systems may result.

IMS/ESA V6 Customization Guide

In Chapter 9, RECON I/O Exit Routine (DSPCEXT0), replace the present contents of the “Attributes of the Routine” section with the following: You must write and link-edit this routine as reentrant. It is entered from DBRC in 31-bit addressing mode and must return to DBRC in 31-bit addressing mode. All parameters and data areas supplied to this exit routine by DBRC are located above the 16-megabyte line. In addition, load module DSPCINT0, in which the routine located, resides above the line. Note that due to the residency of DSPCINT0, unless you specify otherwise, GETMAIN will acquire storage above the 16-megabyte line when issued for the exit routine.

No further calls to the exit routine will occur if it terminates abnormally. DBRC will recover the termination and carry on normally thereafter.

Appendix B The Real Rules of Archive

The archiving rules were introduced for IMS DB/DC in IMS V1.3. The same rules apply to users of DBCTL.

DBRC Records

There are three basic record types that are built in RECON with DBRC at the Log Control level.

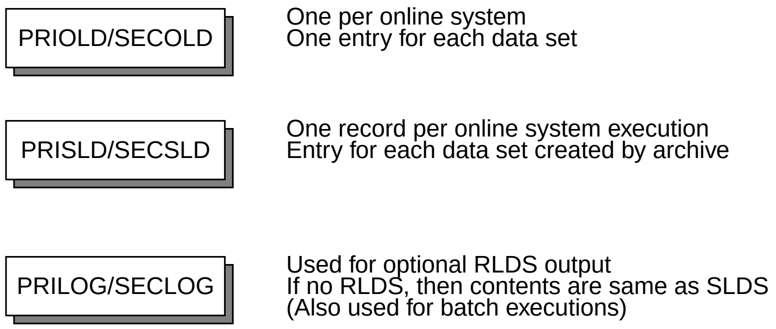

PRIOLD/SECOLD
One per online system
One entry for each data set
PRISLD/SECSLD
One record per online system execution
Entry for each data set created by archive
PRILOG/SECLOG
Used for optional RLDS output
If no RLDS, then contents are same as SLDS
(Also used for batch executions)

Figure 68 DBRC Records for Archive

- PRIOLD (SECOLD)

  There is one PRIOLD (and SECOLD if dual) record for each online IMS system that accesses this RECON pair. This record is reused at each restart.

For the purposes of this document, we shall assume that there is one online IMS system and only one PRIOLD record. In the PRIOLD record there is one entry for each OLDS.

- PRISLD (SECSLD)

  One of these records is created each time online IMS is started. It is used for recording the SLDS output data sets from the archive utility.

- PRILOG (SECLOG)

  One of these records is created each time online IMS is started. It is used for recording the RLDS output data sets from the archive utility. RLDS produced by archiving OLDS contain X '24', '42', '50', '51', '52' and '59' log records. If no records are found for the RLDS, the SLDS information is recorded in the PRILOG record and no RLDS are created.

Note: If you should ever turn DBRC on for batch processing, this record type is also used for recording batch log information. This is confusing. The manuals call batch logs SLDS because they contain all the batch log records and not just database change records. DBRC records them as PRILOGs because this is the record type used for generating JCL.

The NOTIFY.PRILOG command creates information about an RLDS by default. To enter information about an SLDS, you must use NOTIFY.PRILOG SLDS.

NOTIFY.PRILOG for batch logs (normal use of this command) requires SSID(...).

SECLOG and SECSLD records are created by archive if required. It is only then that DBRC can tell if you are producing dual output.

SLDS

SLDSs are produced either as a result of archiving OLDSs or as the output log from a batch job. Having produced an SLDS in batch, you may decide to copy the log data set to a different media. For example, you may log to disk during a batch job and subsequently archive to tape/MSS. This copy can also be performed by the archive utility with appropriate re-blocking and, optionally, under DBRC control.

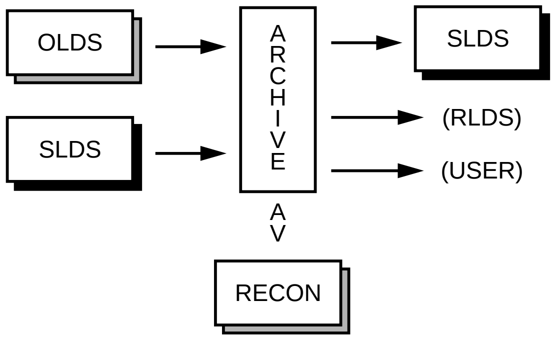


Figure 69 Archive Input and Output

If the batch job is run with DBRC, then run archive with DBRC. You update the PRILOG records with the SLDS you just produced. If you produce an RLDS as well, this is recorded in the PRILOG record. (Batch logs have no PRISLD record).

If the batch job is run without DBRC, run archive without DBRC.

Note: You cannot run backout from an RLDS.

If archiving batch logs, you have to generate the JCL yourself. DFHSM is probably a much easier alternative. See the section on “DBRC and Data Facility Hierarchical Storage Manager (DFHSM)” on page 62 for additional information.

Archive Output (For All Ways of Running)

Archive of online logs must always produce at least one SLDS. Dual SLDS can be produced if desired.

The following record types (at least) may be dropped from the SLDS:10, 18, 45, 48, 5F, 67, 69, A0-FF. Any dropped records are replaced by X'4303' dummy log records in the SLDS to guarantee contiguous log record sequence numbers.

If both primary and secondary SLDSs use cartridges, their contents can be forced to be identical by specifying a FEOV after *n* blocks parameter.

Whenever a system checkpoint log record or an extended checkpoint record is encountered during the archive process, the archive utility produces a message on SYSPRINT to identify the checkpoint log records in the SLDS being created.

If DCB is not coded for output, the default is BUFNO=2, and BLKSIZE and LRECL are coded as input data sets. You are recommended to use large block sizes and at least ten buffers to make it go faster.

Single or dual RLDSs can be produced via control statements.

User data sets can be created in one of two ways:

- control statements—like DFSERA10
- user exits—sample in SPRM

User data sets are not recorded in RECON.

Archiving

The archiving process is used to free up OLDS for reuse by the online IMS system. There are two ways of running the archive process—automatic or manual. The choice is made at system startup through the execution time parameter ARC=nn where *nn* is a number between 0 and 99. 0 means no automatic archive. It can be changed using the operator command /STA AUTOARCH nn.

If you are using automatic archiving, then an archive job is created automatically when one of the following scenarios occur:

- The user-specified number (ARC=nn) of OLDS (pairs) is filled. For example, archive every two OLDS.

  Note: Every time an OLDS switch occurs, a message similar to one of the following is used.

```
DFS3257I ONLINE LOG NOW SWITCHED FROM DFSOLP02 to DFSOLP03
DFS3257I ONLINE LOG NOW SWITCHED FROM DFSOLS02 to DFSOLS03
```

- The OLDS (pair) is closed with a /DBR or /DBD command without NOFEOV, for example, prior to database recovery, or a /SWI OLDS command.
- The OLDS (pair) is closed during IMS termination.
- The OLDS (pair) is closed from the WADS during /ERE.

  Note: Every time an archive job is generated, a message similar to the following is issued.

```
DFS2484I JOBNAME=AR133456 GENERATED BY LOG AUTOMATIC ARCHIVING
```

OLDS Switch

DBRC is notified at the OLDS switch. A sample RECON listing would show the following:

```
PRIOLD
SSID=IMSA           # DD ENTRIES=3
DDNAME =DFSOLP00  DSN=IMS310.OLDSP00
START  =94.128 12:34:56.7      STOP  =94.128 13:34:56.7
STATUS=ARC COMPLETE                  FEOV=NO     AVAIL
PRILOG TIME =94.128 12:34:56:7  ARCHIVE JOB NAME=AR133456

DDNAME =DFSOLP01  DSN=IMS310.OLDSP01
START  =94.128 14:15:16.7      STOP  =94.128 15:16:17.8
STATUS=ARC NEEDED                    FEOV=NO     AVAIL
PRILOG TIME =94.128 14:15:16:7

DDNAME=DFSOLP02  DSN=IMS310.OLDSP02
START = 94.128 15:16:17.8      STOP  =00.000 00:00:00.0
STATUS=ACTIVE                        FEOV=NO     AVAIL
PRILOG TIME =94.128 14:15:16.7
```

Figure 70 Sample RECON Listing for the OLDS Switch

The PRIOLD record contains three DD entries. The first one refers to DFSOLP00 from a previous session, which we will ignore. The second one refers to DFSOLP01—closed, and with ARC NEEDED status. The third one refers to DFSOLP02—now ACTIVE. Stop time for the DFSOLP01 entry is the same as the start time of the DFSOLP02 entry. Both have the same PRILOG START TIME (=timestamp of record where archive output is recorded), so you can see that they come from one contiguous IMS session.

Note: At the OLDS switch time, the next+1 OLDS is pre-opened so that it is immediately ready at the next OLDS switch.

GENJCL.ARCHIVE

If automatic archiving is not chosen, the MTO must create a manual archive job using the GENJCL.ARCHIVE command at user selected intervals.

There are two options available on the GENJCL.ARCHIVE command: archive all OLDS that need archiving (this is the default option and is used by automatic archiving), or archive only specified contiguous OLDSs.

```
GENJCL.ARCHIVE ALL
```

OR

```
GENJCL.ARCHIVE OLDS(ddname1,ddname2,....)
```

If you are archiving to cartridge, the ALL option is recommended. This will pick up any OLDS that were not archived, guaranteeing that all OLDS are archived.

If you are archiving to DASD, you must realize that an out-of-space condition on the output data set will cause the utility to abend. During the pre-installation tasks you calculate the correct size for your DASD SLDS (and RLDS). This is based on a specific number of OLDS being archived together. If archiving to DASD, you should always archive a set number of OLDS. There is a parameter MAXOLDS that limits the number of OLDS to be archived in each job step. How to use this with automatic archiving is covered below.

After /DBR or /SWI OLDS Command

After a /DBR command without the NOFEOV parameter, the current OLDS is closed. The next available OLDS is opened and a SIMPLE checkpoint is written in the new OLDS. The database is stopped and deallocated. For IMS this is a recovery point, so DBRC will keep track of this event in the RECON data set.

IMS will issue the following message:

```
DFS3257I ONLINE LOG NOW SWITCHED - FROM DFSOLP02 TO DFSOLP03
```

At this time, automatic archive is always triggered and all OLDSs needing archive are selected.

A listing of RECON at this stage would show:

```
PRIOLD
SSID=IMSA           # DD ENTRIES=4
DDNAME =DFSOLP00  DSN=IMS310.OLDSP00
...........

DDNAME =DFSOLP01  DSN=IMS310.OLDSP01
START  =94.128 14:15:16.7      STOP  =94.128 15:16:17.8
STATUS=ARC SCHED                      FEOV=NO     AVAIL
PRILOG TIME =94.128 14:15:16:7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP02  DSN=IMS310.OLDSP02
START = 94.128 15:16:17.8      STOP  =94.128 16:17:18.9
STATUS=ARC SCHED                      FEOV=YES    AVAIL
PRILOG TIME =94.128 14:15:16.7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP03  DSN=IMS310.OLDSP03
START = 94.128 16:17:18.9      STOP  =00.000 00:00:00.0
STATUS=ACTIVE                                FEOV=NO   AVAIL
PRILOG TIME =94.128 14:15:16.7
```

Figure 71 Sample RECON Listing After a /DBR Command

In this listing, it is important to note FEOV=YES, which will force an end of output data set at the end of archiving DFSOLP02. This is done to honor the recovery point established by the /DBR ... command. It does this by generating any subsequent OLDS to be archived in a separate job step. (Generating multiple job steps in archive is considered in more detail below.)

Also, the archive job name shows the stop time of the last OLDS being archived.

/SWI OLDS will also switch OLDS and trigger archive, but this is not a recovery point.

After Archive

When the archive job AR161718 starts, the ARCHIVE-STATUS in RECON is changed from SCHEDULED to STARTED.

After normal completion of job AR161718, the situation is:

```
PRILOG
START = 94.128 14:15:16.7      STOP  =00.000 00:00:00.0
SSID=IMSA        #DSN=1         IMS

DSN=IMS310.RLDSP.IMSA.D94128.T1415167.V11
START = 94.128 14:15:16.7      STOP  =94.128 16:17:18.9
FILE SEQ=0001   #VOLUMES=0001    -VOLSER-    -STOPTIME-
                                 RLDS02  94.128 16:15:14.3
---------------------------------------------------------------
PRISLD
START = 94.128 14:15:16.7      STOP  =00.000 00:00:00.0
SSID=IMSA        #DSN=1         IMS

DSN=IMS310.SLDSP.IMSA.D94128.T1415167.V11
START = 94.128 14:15:16.7      STOP  =94.128 16:17:18.9
FILE SEQ=0001   #VOLUMES=0001    -VOLSER-    -STOPTIME-
                                 SLDS02  94.128 16:15:14.3
------------------------------------------------------------
PRIOLD
SSID = IMSA          # DD ENTRIES=4
DDNAME =DFSOLP00  DSN=IMS310.OLDSP00
....

DDNAME =DFSOLP01  DSN=IMS310.OLDSP01
START  =94.128 14:15:16.7      STOP  =94.128 15:16:17.8
STATUS=ARC COMPLETE                   FEOV=NO     AVAIL
PRILOG TIME =94.128 14:15:16:7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP02  DSN=IMS310.OLDSP02
START = 94.128 15:16:17.8      STOP  =94.128 16:17:18.9
STATUS=ARC COMPLETE                   FEOV=YES    AVAIL
PRILOG TIME =94.128 14:15:16.7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP03  DSN=IMS310.OLDSP03
START = 94.128 16:17:18.9      STOP  =00.000 00:00:00.0
STATUS=ACTIVE                                FEOV=NO   AVAIL
PRILOG TIME =94.128 14:15:16.7
```

Figure 72 Sample RECON Listing After Archive Job AR161718

One entry was created in PRILOG/PRISLD to record the first archived data set. START time is the open time of the first OLDS archived (DFSOLP01). STOP time is the close time of the last OLDS archived (DFSOLP02). STOPTIME is the timestamp of the last database update record contained in this volume.

PRIOLD entries referring to DFSOLP01 and DFSOLP02 ddnames are updated in order to keep track of their archiving.

End of Session

If you now close down IMS using, for example, /CHE DUMPQ, then the OLDS is closed and automatic archive triggered.

Having closed down IMS and run the archive, RECON now shows the following:

```
PRILOG
START = 94.128 14:15:16.7      STOP  =94.128 18:01:02.3
SSID=IMSA        #DSN=2         IMS

DSN=IMS310.RLDSP.IMSA.D94128.T1415167.V11
START = 94.128 14:15:16.7      STOP  =94.128 16:17:18.9
FILE SEQ=0001   #VOLUMES=0001    -VOLSER-    -STOPTIME-
                                 RLDS02  94.128 16:15:14.3
DSN=IMS310.RLDSP.IMSA.D94128.T1617189.V11
START = 94.128 16:17:18.9      STOP  =94.128 18:01:02.3
FILE SEQ=0001   #VOLUMES=0001    -VOLSER-    -STOPTIME-
                                 RLDS03  94.128 17:59:01.2

--------------------------------------------------------------
Ditto for SLDS

---------------------------------------------------------------
PRIOLD
SSID = IMSA          # DD ENTRIES=4
DDNAME =DFSOLP00  DSN=IMS310.OLDSP00
....

DDNAME =DFSOLP01  DSN=IMS310.OLDSP01
START  =94.128 14:15:16.7      STOP  =94.128 15:16:17.8
STATUS=ARC COMPLETE                   FEOV=NO     AVAIL
PRILOG TIME =94.128 14:15:16:7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP02  DSN=IMS310.OLDSP02
START = 94.128 15:16:17.8      STOP  =94.128 16:17:18.9
STATUS=ARC COMPLETE                   FEOV=YES    AVAIL
PRILOG TIME =94.128 14:15:16.7  ARCHIVE JOB NAME=AR161718

DDNAME=DFSOLP03  DSN=IMS310.OLDSP03
START = 94.128 16:17:18.9      STOP  =94.128 18:01:02.3
STATUS=ARC COMPLETE                          FEOV=NO   AVAIL
PRILOG TIME =94.128 14:15:16.7  ARCHIVE JOB NAME=AR180102
```

Figure 73 Sample RECON Listing at the End of a Session

A Recent Archive Hiccough

Customer was trying to delete PRIOLD and SECOLD records from RECON (as part of DR cleanup) and discovered that the SECOLD records wouldn't go when the PRIOLD records were deleted. Further investigation showed that the timestamps in the PRIOLD and SECOLD records were not the same.

Orphan SECOLD records on the RECON can be caused by going from dual logging to single logging, and not deleting the PRIOLD records in the process. Unfortunately, the manual doesn't tell you to do a DELETE.LOG when going from dual to single, but IBM is working on that. If you don't do the DELETE.LOG, and if you cycle IMS enough times to go through all the OLDS data sets, you will end up with SECOLD records that have no matching PRIOLD records in terms of timestamp and log sequence number, i.e., an orphan SECOLD.

What Does Archive Select?

Automatic archive ignores any OLDS in ARCHIVE STARTED status. This is to avoid the problem of two jobs trying to archive the same OLDS. Assume OLDS0 fills and job ARC0 is generated and started. If OLDS1 now fills, a new job will be generated (ARC1) just containing OLDS1. If job ARC0 fails, DBRC will not include OLDS0 in any automatic archive until either a CHANGE.PRILOG OLDS(ddname) ARNEEDED is issued, or until IMS is restarted. (IMS restart resets any ARCHIVE STARTED to ARCHIVE NEEDED.)

You cannot change an OLDS from ARCHIVE COMPLETE to ARCHIVE NEEDED and rerun archive—DBRC will not allow this. Some customers have tried to do this because they had an unreadable RLDS and tried to re-archive the OLDS to produce a new RLDS. The correct way to handle this situation is to run archive with PARM='DBRC=N' with the OLDS entered on the input SLDS DD statement as if the OLDS were a batch log created outside DBRC or to follow the procedures given in “How to Use a SLDS in Place of a Bad RLDS” on page 69.

Parallel Archives

One method of running archives in parallel is to manually generate archive JCL using GENJCL.ARCHIVE OLDS(OLDSname). Some customers have done this using AOI to trigger the GENJCL. Since each archive job now archives a different OLDS, you can run them in parallel.

There is a parameter to help you—MAXOLDS. This limits the number of OLDSs in each step of the archive job, e.g., MAXOLDS(1). This parameter is not used by automatic archiving. How to overcome this is shown below.

Note: Parallel archive won't work if all the jobs try to write to the same GDG.

Multi-Step Archives

It is quite easy to create multi-step archive jobs. For example, the /DBR command is used (see above), or the OLDS being archived are not contiguous. The problem here is that when using GDGs, DBRC generates the same DD output statements in each step, e.g., GDG(+1). One technique to overcome this is to use the %SET command. See Appendix C, "DBRC GENJCL Enhancements." The first step uses archive JCL with GDG(+1). At the end of the skeletal JCL there is a %SET command pointing at a second member that has GDG(+2) coded, etc.

Auto Archive with MAXOLDS

There is a technique that allows you to use automatic archiving with the MAXOLDS parameter. This is designed for those occasions where the archives have stacked up, and if you ran automatic archiving, it would create one giant multi-volume SLDS. If you were archiving to DASD, it would probably abend B37.

The technique replaces the archive skeletal JCL (ARCHJCL) with a GENJCL.ARCHIVE job so that whenever automatic archiving gets triggered it runs your GENJCL.ARCHIVE job. The command used in the skeletal JCL is GENJCL.ARCHIVE ALL MAXOLDS(1) MEMBER(REALARCH) SSID(%SSID), where a second skeletal JCL member REALARCH contains the normal archive JCL. If there is more than one OLDS to archive, this produces multiple archive jobs/job steps (depending on JOB/NOJOB) so there is no problem with GDGs.

In all cases, you should run the archive job(s) non-swappable.

Appendix C DBRC GENJCL Enhancements

Additional keywords were added to the DBRC GENJCL function many years ago via APAR PP21253 to provide improved support for generating multi-step jobs in utility functions such as archive or change accumulate.

Note: This chapter/APAR was written prior to the availability of Systems Managed Storage (SMS), and is aimed at an environment where the catalog is updated at the end of job (not end of step) for GDGs.

PP21253 APAR Text

The following text is basically as it appeared in RETAIN.

Problem

After applying the multiple execution of change accumulation APAR, the second and subsequent job steps generated by issuing a GENJCL.CA command may contain invalid JCL statements and/or may generate incorrect change accumulation utility JCL, depending on how the skeletal JCL CA member has been tailored in the installation. As a general rule, replacing DBRC %KEYWORDS with hard-coded JCL statements will fail when multiple steps are generated since the JCL is simply duplicated for each subsequent step.

This makes it difficult to handle GDGs or situations where the output from one step is passed as input to the next step and non-standard data set names are used.

Conclusion

This APAR fix corrects some problems that occur when multiple steps are generated. It changes the manner in which the optional parameters JOB/NOJOB are handled so that multiple job steps are consistent. It also introduces new capabilities that allow considerably more versatility when generating JCL—especially for multiple steps.

- Currently, when VOLLIST and VOLNUM parameters are specified on a GENJCL.CA command, the same list of volume serial numbers is used for each step generated by the command. Subsequent steps write over the change accumulation output data set created by the previous step. This APAR generates only one step when the VOLLIST parameter is specified, even if a VOLNUM parameter would have caused several steps to be generated.

- Currently, the GENJCL command processor unconditionally sets the NOJOB parameter after JCL has been produced from one pass through the skeletal JCL member. This is to prevent the job card from being reproduced if more steps are generated due to the VOLNUM parameter being specified. If the skeletal JCL member has been set up to contain more than one job, on the first pass through all jobs JCL will be generated if the parameter JOB is specified (JOB is the default). However, for subsequent steps, DBRC will strip out all job statements encountered in the skeletal JCL member since NOJOB has been set on by the GENJCL command processor.

  This APAR changes this, so the setting of the JOB/NOJOB parameter can be maintained. For subsequent steps, the first job statement encountered in the skeletal JCL member prior to finding an EXEC statement is stripped off. Thereafter, job statements are treated according to the setting of the JOB/NOJOB parameter. This is done to ensure that subsequent steps are generated consistently with the first.

- Three DBRC reserved keywords are introduced to allow for more versatility. They are %STEPNO, %CNTR, and %DATE.

  %DATE is replaced with the date when the GENJCL command is issued in the format of *yyddd*, where *yy* is the year and *ddd* is the day (for example 84102). %DATE may be specified any number of times in a skeletal JCL member.

%STEPNO is a five-character counter initialized to zero and incremented by one for each occurrence of the keyword prior to replacement. It is reset to zero for each GENJCL command that specifies the JOB parameter. %STEPNO is a keyword that can be used any number of times within a skeletal JCL member. The maximum value of this keyword is 32768, after which it is set back to zero. Leading zeros are stripped off before replacement. This keyword is intended to provide a facility for numbering the steps produced by a GENJCL command (and thus provide for unique step names) although not limited to this purpose. Because it is incremented first, the value 0 never replaces the keyword.

%CNTR is a five character counter initialized to zero and incremented by one for each occurrence of the keyword prior to replacement. It is set back to zero each time a GENJCL command is issued and whenever a JOB statement is reproduced from the skeletal JCL member. The maximum value of this keyword is 32768, after which it is set back to zero. Leading zeros are stripped off before replacement. This keyword may be used any number of times in a skeletal JCL member. Because it is incremented first, the value 0 never replaces the keyword.

This is a general purpose keyword that differs from the %STEPNO keyword in how and when is it initialized to zero.

- To provide more versatility and capability, several keywords have been given a new multiple-use attribute that allows them to be used as often as desired in a skeletal JCL member.

  These keywords are %RCNDSN1, %RCNDSN2, (and %RCNDSN3 for release 3), %TIME, %DATE, %STEPNO, and %CNTR. These keyword values either do not change or are expected to change, which allows them to be used multiple times. No other DBRC reserved keywords can be assigned this attribute. The time and date are obtained when a GENJCL command is issued, and they remain constant for the execution of the command. User keywords (defined by the user keys parameter on the GENJCL command) that specify a DBRC keyword as the substitution source are assigned the same multiple use attribute as the substitution source. Thus, a user keyword that specifies %CNTR as the substitution source may also be used any number of times in a skeletal JCL member.

  The replacement value for a user keyword that has %STEPNO or %CNTR specified as the substitution source is the current value without incrementing it (which is zero for the initial value).

- A new DBRC reserved control word allows the specification of a different JCL skeletal member to be named when multiple steps are being produced (such as GENJCL.CA with a VOLNUM parameter supplied). The format is:

  ```
  %set member = newmembername
  ```

  The value *newmembername* is the name of a skeletal JCL member to be used for the next pass generating JCL. This member must reside in the library named in the JCLPDS DD statement.

  %SET must begin in column one. At least one blank must delimit each word, and the complete statement must not extend past column 72. This statement can occur anywhere within the current member, but will not take effect until processing of the current member is complete. If the statement is encountered more than once, the last occurrence will be the name used. If desired, the new member may also contain a %SET statement specifying any member name. The name of the new member is not used until processing of that member begins. Thus it is possible to specify an incorrect member name and not have any error condition occur until a GENJCL command is issued. This causes sufficient steps to be generated and the member to be read.

Appendix D IMS Change Accumulation Utility

The following definitions will help in understanding the rules.

CA Group

When using DBRC with CA, database data sets are placed in CA groups. An execution of the CA utility processes all of the data sets in one and only one CA group. Data sets belong to only one CA group.

CA Stop Time

The stop time of a CA execution is the latest stop time for any input log.

Allocated

A database data set is considered to be allocated to a subsystem from the time of the first update to the data set following its authorization to the subsystem until the unauthorization of the data set.

Purge Time

The purge time is a time specified in the inputs to the CA utility. Database change records previous to this time are not accumulated. Each database data set may have its own purge time. The DBRC GENJCL.CA command will cause appropriate purge times to be included in the CA input.

An input log is considered to have updates for a data set if its start and stop times span a time during which the database data set was allocated to the subsystem and the purge time for the data set is not later than the stop time for the log.

The change accumulation utility will execute even though its inputs are an incomplete subset of logs. The DBRC GENJCL.CA command processing has been changed to create CA JCL which includes an incomplete set of logs.

To allow the processing of incomplete subsets, the change accumulation utility has added a new type of output record. This is the spill record. It contains information about database changes that come from logs in the incomplete subset. This information cannot be included in the old style accumulation records. These accumulation records do not maintain the timestamps of database changes from the original logs. Since some of the missing logs may have updates which both precede and follow the times of the updates supplied to CA, these must be maintained to determine whether and how to apply the changes during a later CA execution. The spill records maintain these timestamps with the changes. The spill records for a database data set block immediately follow the accumulation record, if any, for the block in the CA output. The CA output is sorted by block or CI location within the data set. The spill records will be interleaved in the accumulation records.

A “netting up” process is used to limit the amount of information in spill records for one block. Only the latest change for any portion of a database data set block is kept in a spill record. If multiple change records alter the same part of a block, the information about the earlier change is discarded. A spill record will never contain information about two changes to the same byte in a block. This is similar to the process used for accumulation records which has always been done. The difference is that the timestamps are maintained for each portion in the spill record.

Changes from logs which are part of the incomplete subset are never included in accumulation records. Instead, they are included in spill records.

These changes may cause the size of CA output files to grow. When incomplete log subsets are used as input, the output files may contain both accumulation and spill records for the same database blocks. The accumulation process for accumulation records and the netting up process for spill records ensure that there are not multiple accumulation records or multiple spill records for changes to the same bytes of any blocks. This will limit the size of output files.

Database Recovery Utility

The CA utility can now be run with fewer restrictions, but it does not change the restrictions on running the database recovery utility. A recovery point for the database must be created in order to recover a database data set. A recovery point is a time when a complete subset of logs exists for the database data set. CA outputs with spill records for a database data set cannot be used to recover the database data set.

The database recovery utility has been changed to recognize spill records. The presence of spill records indicates that the CA utility was run against an incomplete set of logs and the CA output is invalid for recovering the database data set. When the database recovery utility recognizes a spill record for the data set being recovered, it issues a DFS0747I message. The text of the message is:

```
DFS0747I CHANGE ACCUM DATA SET IS MARKED INCOMPLETE
```

If DBRC is invoked for the database recovery utility execution, the presence of spill records will cause the utility to fail during initialization. It will never attempt to read the CA data set. As explained below, DBRC keeps track of the presence of incomplete log subsets and uses this information to inform the database recovery utility that a successful recovery cannot be done.

DBRC

DBRC will allow GENJCL.CA and CA utility execution with an incomplete subset of logs. It will prevent execution of the database recovery utility with a CA output which contains spill records for the data set to be recovered.

The LIST.CAGRP command has been enhanced to indicate the status of each data set in the group. It reports which data sets in the group have spill records and which do not. Those without spill records are indicated as being complete, that is, YES appears in the CMP column.

DBRC must be used with the CA utility in order for it to create spill records. DBRC informs the utility about the presence of incomplete log subsets for a database data set. If they exist, CA creates spill records for changes from the logs in the incomplete subset. It is possible that some of the logs used as input will be from complete subsets and some from the incomplete subset. Thus, some records may be accumulated and some spilled. If an incomplete subset exists for a database data set, DBRC will pass the data set sequence number (DSSN) for the incomplete subset to the utility. CA will accumulate the records from logs with lower DSSN values and spill the records from the remaining logs.

The decision about complete versus incomplete log subsets is made for each database data set. It is possible that the logs used as input to the CA utility will be complete for some database data sets and incomplete for others.

The NOTIFY.CA command has been changed to allow the specification of COMPLETE or INCOMPLETE for each database data set in the CA group.

Appendix E Acronyms and Abbreviations

ADS	Area Data Set
AOI	Automated Operator Interface
APAR	Authorized Program Analysis Report
APPLID	Application Identifier
BCS	Basic Catalog Structure
BLS	Block Level Sharing
BMP	Batch Message Processing
CA	Change Accumulation
CAGRP	Change Accumulation Group
CATDS	Cataloged Data Set
CI	Control Interval
CIC	Concurrent Image Copy
CICS	Customer Information Control System
CPU	Central Processing Unit
DASD	Direct Access Storage Device
DB	Database

DB/DC	Database/Data Communication
DBCTL	Database Control
DBD	Database Definition
DBRC	Database Recovery Control
DD	Data Definition
DEDB	Data Entry Database
DFHSM	Data Facility Hierarchical Storage Manager
DL/I	Data Language I
DLI	Data Language Interface
DSLOG	Data Set Log
DSSN	Data Set Sequence Number
EEQE	Extend Error Queue Element
EQE	Error Queue Element
GDG	Generation Data Group
GENJCL	Generate JCL
HIDAM	Hierarchical Indexed Data Access Method
HISAM	Hierarchical Indexed Sequential Access Method
I/O	Input/Output
IC	Image Copy
ICF	Integrated Catalog Facility
IMS	Information Management System
IMS/CICS	Information Management System/Customer Information Control System
IMS/ESA	Information Management System/Enterprise Systems Architecture
IMS/VS	Information Management System/Virtual Storage

IRLM	IMS/VS Resource Lock Manager
JCL	Job Control Language
JES	Job Entry Subsystem
KSDS	Key Sequenced Data Set
LPI	Load Program Interface
LSN	Lock Sequence Number
LSR	Local Shared Resources
MADS	Multiple Area Data Sets
MPP	Message Processing Program
MSDB	Main Storage Database
MSS	Mass Storage System
MTO	Master Terminal Operator
MVS	Multiple Virtual Storage
NOCATDS	Non-Cataloged Data Set
OIC	Online Image Copy
OLDS	Online Log Data Set
OS	Operating System
OSAM	Overflow Sequential Access Method
PDS	Partitioned Data Set
PL/1	Programming Language One
PRILOG	Primary Recovery Log
PRISLD	Primary System Log Data Set
PSB	Program Specification Block
PTF	Program Temporary Fix

RACF	Resource Access Control Facility
RECON	Recovery Control (data set)
RECOVCTL	Recovery Control
REORG	Reorganization
RLDS	Recovery Log Data Set
SECLOG	Secondary Recovery Log
SECSLD	Secondary System Log Data Set
SHARECTL	Share Control
SLDS	System Log Data Set
SSID	Subsystem Identifier
SUBSYS	Subsystem
TOD	Time Of Day
UIC	User Image Copy
VOLSER	Volume Serial (number)
VS	Virtual Storage
VSAM	Virtual Storage Access Method
VTAM	Virtual Telecommunications Access Method
VTOC	Volume Table of Contents
VVDS	Virtual Volume Data Set
WTOR	Write-to-Operator with Reply

Index

Notes

103696