



# **Implementing Hardware Data Compression Using Database Director**

Copyright © 2005 NEON Enterprise Software, Inc. All rights reserved.

The symbols ® and ™ denote USA trademark rights.

NEON Mission Control is a trademark of NEON Enterprise Software. Affinities Server, NEON 24X7, NEON iBuild, NEON iChange, NEON iCheck, NEON iCopy, NEON iLoad, NEON iRecover, NEON iUnload, Partitioned Database Facility, and PDF are trademarks of NEON Systems, Inc., in the USA and in other select countries, and are licensed to NEON Enterprise Software.

All other trademarks are the property of their respective owners.

# Table of Contents

Introduction.....	4
What To Do With the Data .....	4
Is HDC Really Free?.....	4
So Much Data, So Little Time .....	5
Technical Overview .....	6
Implementation .....	7
1. Define the database definition (DBD) .....	7
2. Create an unload file(s) of the target database.....	8
3. Hardware Data Compression Data Analyzer Utility .....	9
4. Modify the database definition (DBD) .....	11
Summary .....	14
Shortening the Process.....	15
Versatility and Availability.....	15

# Introduction

Database Director is a powerful data management tool that automates the tasks associated with reorganizing a database. Database Director has the capability of managing several tasks while performing an online reorganization – such as space management, data extraction, and hardware data compression. This document explains how to implement hardware data compression (HDC) using Database Director.

This document assumes that you understand the basic concepts of IMS database administration and hardware data compression.

## What To Do With the Data

As data retention requirements grow, so do the size of your databases. Unfortunately, there are limitations to the size of database datasets – 4GB for VSAM and 8GB for OSAM. If you need more space, you have the following options:

- Multiple dataset groups. This option allows you to spread database segments into as many as 10 different datasets.
- Database Partitioning. This requires that you convert your database to a partitioned database such as NEON Enterprise Software's Partition Database Facility or IBM's High Availability Large Database (HALDB).
- Database segment compression. This solution requires that you specify a segment compression routine in the database definition (DBD).
- Convert from VSAM to OSAM

Hardware compression is one of several solutions that can provide additional capacity in a database dataset. Hardware compression is also an alternative to a software compression tool, whether it is a vendor-supplied or internally developed tool. You can easily implement hardware compression for your IMS databases using Database Director.

## Is HDC Really Free?

The process of implementing HDC is very simple. Take an unload file, run it through the HDC data analyzer utility, modify your DBD, reorganize the database – unload with the old DBD, load with the new DBD. Simple isn't it... or is it?

First, you must understand that your best compression algorithms will be implemented if the HDC data analyzer utility can produce data compression routines for each database segment. This means that you cannot simply use your latest unload file. You need to break out each segment into a separate sequential file.

Second, how do you create an unload file? It's obvious that you would have one when you next reorganize the database. Unfortunately, you may want (or need) to implement data compression during the next reorganization window. Unloading the database and building the requisite files for the HDC data analyzer would simply elongate the reorganization outage.

Traditionally, the price of implementing HDC has been the significant effort required creating the compression routine and the online outage required to implement the routine. With Database

Director Persist and Eclipse iExtract, you can perform all the required tasks with very little database outage – the time it takes to issue some IMS commands.

## **So Much Data, So Little Time**

Database Director Persist implements a new compression routine without the lengthy outage associated with the traditional implementation methods. Your database outage can be limited to the time it takes to issue a couple of IMS commands.

After Database Director Persist completes the reorganization process, it can restart (/DBR) the database in preparation of implementing the new one. At this time, you can issue the requisite /MOD PREPARE and /MOD COMMIT commands to make the new DBD available to your online IMS system. Afterwards, simply restart the database and your online transactions are flowing again.

Running BMP, DLI and DBB jobs won't even need to be stopped. Database Director Persist can suspend them long enough for you to issue those requisite IMS commands. You can simply resume any suspended jobs and the new hardware data compression is implemented. There is no need to neither terminate the application program nor schedule the implementation of your new hardware data compression routines around production job streams.

# Technical Overview

The technical solution described in this document requires the use of the following products from NEON Enterprise Software:

- **Mission Control.** This is a Sysplex and Web enabled IMS database administration workbench. This product collects the online IMS database updates and passes them to the Database Director Persist job where they will be applied to the newly reorganized database.
- **Database Director.** This product provides a single-step reorganization of any database. It has the ability to reorganize offline or online IMS databases. It is able to reorganize or clone an online IMS database, with full data integrity and maintaining 100% application availability.
- **Eclipse iUnload.** Database Director uses this product to unload the IMS database and produce an unload file. This product can be used in a batch mode outside of Database Director.
- **Eclipse iLoad.** Database Director uses this product to load the new IMS database. This product can be used in a batch mode outside of Database Director.
- **Eclipse iBuild.** Database Director uses this product to build any primary and secondary index IMS databases. This product can be used in a batch mode outside of Database Director.
- **Eclipse iCopy.** Database Director uses this product to build an Image Copy of the IMS database and its indexes. This product can be used in a batch mode outside of Database Director.
- **Eclipse iExtract.** By substituting this product for the Eclipse iUnload product during database cloning, Database Director can produce unload files of each database segment into separate files for the Hardware Data Compression data analyzer utility.
- **Prefix Update.** Database Director uses this product to update the logical relationship prefixes within a database. This product can be used in a batch mode outside of Database Director.

All of these products are seamlessly integrated by Database Director to provide a one-step online reorganization of an IMS database.

# Implementation

This section describes the steps required for implementing hardware data compression.

## 1. The database definition (DBD)

```
DBD          NAME=DDAHODB,
              ACCESS=(HDAM,OSAM),
              RMNAME=(DFSHDC40,4,200,)

DSG001 DATASET DD1=DDADD001,
              SIZE=(512),
              SCAN=3,
              FRSPC=(0,5)

SEGM        NAME=DDAROOT,
              PARENT=0,
              BYTES=100,
              PTR=TB
              FIELD NAME=(DDAKEY,SEQ,U),
                  START=1,
                  BYTES=10,
                  TYPE=C

SEGM        NAME=DDANASEG,
              PARENT=(DDAROOT),
              BYTES=300,
              PTR=TB
              FIELD NAME=(DDANAME,SEQ),
                  START=1,
                  BYTES=50,
                  TYPE=C

              FIELD NAME=/SX1
              LCHILD NAME=(SECX01,DDAHOIDB),PTR=INDX
                  XDFLD NAME=NAMINDX,SEGMENT=DDANASEG,
                      SRCH=(DDALNAME,DDAFNAME,DDAMINIT),
                      SUBSEQ=/SX1

SEGM        NAME=DDATRSEG,
              PARENT=(DDAROOT),
              BYTES=100,
              PTR=TB
              FIELD NAME=(DDATRTP),
                  START=1,
                  BYTES=5,
                  TYPE=C

DBDGEN
FINISH
END
```

As you can see, this is a fairly simple database. Three database segments with a secondary index on the name/address segment. We are going to add a compression routine to all three segments.

## 2. Create an unload file(s) of the target database

First, create unload files for each segment type in the database. To accomplish this, use the cloning feature of Database Director along with Eclipse iExtract to split each segment type into three different datasets.

The cloning feature of Database Director allows you to clone a copy of the online database with full pointer integrity and without any disruption to online transaction processing.

```
//          <<< Your Job Card Goes Here >>>
//*****
//*          DATABASE DIRECTOR (D2)
//*****
//D2CLONE EXEC PGM=NSOMAIN,REGION=0M,DYNAMNBR=250
//STEPLIB DD DISP=SHR,DSN=NEON.LOAD
//          DD DISP=SHR,DSN=IMS.SDFSRESL
//          DD DISP=SHR,DSN=IMS.USEREXIT
//IMSDALIB DD DISP=SHR,DSN=IMS.MDALIB
//          DD DISP=SHR,DSN=IMS.SDFSRESL
//IMS      DD DISP=SHR,DSN=IMS.PSBLIB
//          DD DISP=SHR,DSN=IMS.DBDLIB
//DDAROOT DD DSN=DDAROOT.UNLOAD,
//          DISP=(,CATLG),
//          UNIT=SYSALLDA,SPACE=(CYL,(10,1)),
//          DCB=(DSORG=PS,LRECL=27994,BLKSIZE=27998,RECFM=VB)
//DDANASEG DD DSN=DDANASEG.UNLOAD,
//          DISP=(,CATLG),
//          UNIT=SYSALLDA,SPACE=(CYL,(10,1)),
//          DCB=(DSORG=PS,LRECL=27994,BLKSIZE=27998,RECFM=VB)
//DDATRSEG DD DSN=DDATRSEG.UNLOAD,
//          DISP=(,CATLG),
//          UNIT=SYSALLDA,SPACE=(CYL,(10,1)),
//          DCB=(DSORG=PS,LRECL=27994,BLKSIZE=27998,RECFM=VB)
//NSXUNLD DD *
EXTRACT (DBDNAME (DDAHODB)
        OUTPUTFILE (DDNAME (DDAROOT)
                    SELECT SEGMENT (DDAROOT) )
        OUTPUTFILE (DDNAME (DDANASEG)
                    SELECT SEGMENT (DDANASEG) )
        OUTPUTFILE (DDNAME (DDATRSEG)
                    SELECT SEGMENT (DDATRSEG) ) )
//NSOSYSIN DD *
CLONE DBD (DDAHODB) -
        DSNMASK (PROD.** ,TEST.** ) -
        UNLOAD DELETEDB
//
```

### 3. Hardware Data Compression Data Analyzer Utility

Now that the database segments are split into multiple unload files, run the Hardware Data Compression Data Analyzer utility and build a data compression routine. Below is a JCL proc that will run all the appropriate steps:

```
//HDCXBLD  PROC RESLIB=,
//          EXITLIB=,
//          EXITNM=,
//          UNLOAD=
//*****
//* CREATE STATISTICS AND HDC DICTIONARY OBJECT FILE.          *
//*****
//HDCDGEN  EXEC PGM=DFSZLDU0,REGION=4M,PARM=DFSZHDCD,COND=(0,NE)
//STEPLIB  DD DISP=SHR,DSN=&RESLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//HDCDIN   DD DISP=SHR,DSN=&UNLOAD
//HDCDIT   DD DUMMY
//HDCDOUT  DD DISP=(,PASS),DSN=&&OBJ,
//          UNIT=SYSALLDA,SPACE=(CYL,(100,10),RLSE),
//          DCB=(LRECL=80,BLKSIZE=3120,RECFM=FB,DSORG=PS)
//HDCDCTL  DD DUMMY,DCB=BLKSIZE=80
//SYSLMOD  DD DISP=(,PASS,DELETE),DSN=&&DICTLIB,
//          UNIT=SYSALLDA,SPACE=(CYL,(10,1,5),RLSE),
//          DCB=(RECFM=U,LRECL=0,BLKSIZE=32760,DSORG=PO)
//*****
//* CREATE LOAD MODULE FROM DICTIONARY OBJECT TEXT DECK      *
//*****
//LINK1    EXEC PGM=IEWL,COND=(0,NE),
//          PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD DISP=(OLD,PASS),DSN=&&DICTLIB(DFSZHDCD)
//SYSUT1   DD UNIT=SYSALLDA,DISP=(,DELETE),
//          SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DISP=(OLD,DELETE),DSN=&&OBJ
//*****
//* CREATE THE COMPRESSION EXIT                              *
//*****
//LINK2    EXEC PGM=IEWL,COND=(0,NE),
//          PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD DISP=SHR,DSN=&EXITLIB.(EXITNM)
//SYSUT1   DD UNIT=SYSALLDA,DISP=(,DELETE),
//          SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD SYSOUT=*
//SDFSRESL DD DISP=SHR,DSN=&RESLIB
//DICTLIB  DD DISP=(OLD,DELETE),DSN=&&DICTLIB
```

Now simply execute the data analyzer JCL proc for each unload file and generate a compression exit.

**Note:** When generating compression routine module names, include some type of version or index number in the name. Database Director will not recognize a compression routine change without changing the compression routine name. This will prevent Database Director from performing the requisite tasks required to implement your new compression routine during the online reorganization process.

```
//          <<< Your Job Card Goes Here >>>
//***
//***      Build DDAROOT compression exit - DDARTCXT
//***
//HDCROOT  EXEC HDCXBLD,
//          RESLIB='IMS.RESLIB',           <=== IMS RESLIB
//          EXITLIB='IMS.USEREXIT',        <=== COMPRESSION EXIT LIBRARY
//          EXITNM='DDARTCV1',            <=== COMPRESSION EXIT NAME
//          UNLOAD='DDAROOT.UNLOAD'       <=== DATABASE UNLOAD FILE
//LINK2.SYSLIN DD *
//          INCLUDE DICTLIB(DFSZHDCD)
//          INCLUDE SDFSRESL(DFSZLDX0)
//          PAGE    DFSZHDCD
//          ENTRY   DFSZLDX0
//***
//***      Build DDANASEG compression exit - DDANACXT
//***
//HDCNASEG EXEC HDCXBLD,
//          RESLIB='IMS.RESLIB',           <=== IMS RESLIB
//          EXITLIB='IMS.USEREXIT',        <=== COMPRESSION EXIT LIBRARY
//          EXITNM='DDANACV1',            <=== COMPRESSION EXIT NAME
//          UNLOAD='DDANASEG.UNLOAD'      <=== DATABASE UNLOAD FILE
//LINK2.SYSLIN DD *
//          INCLUDE DICTLIB(DFSZHDCD)
//          INCLUDE SDFSRESL(DFSZLDX0)
//          PAGE    DFSZHDCD
//          ENTRY   DFSZLDX0
//***
//***      Build DDATRSEG compression exit - DDATRCXT
//***
//HDCRSEG  EXEC HDCXBLD,
//          RESLIB='IMS.RESLIB',           <=== IMS RESLIB
//          EXITLIB='IMS.USEREXIT',        <=== COMPRESSION EXIT LIBRARY
//          EXITNM='DDATRCV1',            <=== COMPRESSION EXIT NAME
//          UNLOAD='DDATRSEG.UNLOAD'      <=== DATABASE UNLOAD FILE
//LINK2.SYSLIN DD *
//          INCLUDE DICTLIB(DFSZHDCD)
//          INCLUDE SDFSRESL(DFSZLDX0)
//          PAGE    DFSZHDCD
//          ENTRY   DFSZLDX0
//
```

## 4. Modify the database definition (DBD)

Modify the DBD to make use of the segment compression exit routines. It is important to note that Hardware Data Compression requires that you specify the "INIT" parameter.

```
DBD      NAME=DDAHODB,
        ACCESS=(HDAM,OSAM),
        RMNAME=(DFSHDC40,4,200,)

DSG001 DATASET DD1=DDADD001,
        SIZE=(512),
        SCAN=3,
        FRSPC=(0,5)

SEGM     NAME=DDAROOT,
        COMPRTN=(DDARTCV1,DATA,INIT),
        PARENT=0,
        BYTES=100,
        PTR=TB
        FIELD NAME=(DDAKEY,SEQ,U),
        START=1,
        BYTES=10,
        TYPE=C

SEGM     NAME=DDANASEG,
        COMPRTN=(DDANACV1,DATA,INIT),
        PARENT=(DDAROOT),
        BYTES=300,
        PTR=TB
        FIELD NAME=(DDANAME,SEQ),
        START=1,
        BYTES=50,
        TYPE=C

        FIELD NAME=/SX1
        LCHILD NAME=(SECX01,DDAHOIDB),PTR=INDX
        XDFLD NAME=NAMINDX,SEGMENT=DDANASEG,
        SRCH=(DDALNAME,DDAFNAME,DDAMINIT),
        SUBSEQ=/SX1

SEGM     NAME=DDATRSEG,
        COMPRTN=(DDATRCV1,DATA,INIT),
        PARENT=(DDAROOT),
        BYTES=100,
        PTR=TB
        FIELD NAME=(DDATRTP),
        START=1,
        BYTES=5,
        TYPE=C
```

Now you are ready to run the DBD gen. In order to accommodate Database Director, generate the DBD into a "new" DBD library. Database Director will unload with the "old" DBD and load with the "new" DBD when there are DBD changes detected. To support this functionality, perform both the "new" DBD gen as well as the online reorganization in the same job.

You probably already have JCL procs that you run to generate DBDs. The example below will generate the DBD into a temporary load library for use by the online reorganization.

```

//NEOND2 PROC MBR=,RESLIB=,EXITLIB=,PSBLIB=,NEONLIB=,
//          IMSMAC=,SRCLIB=,DBDLIB=,MDALIB=,TMPDBD=
//*****
//*        DELETE TEMPORARY DATASETS
//*****
//BR14 EXEC PGM=IEFBR14
//SYSLMOD DD DISP=(MOD,DELETE),DSN=&TMPDBD,
//          UNIT=SYSALLDA,SPACE=(TRK,1)
//*****
//*        DEFINE TEMPORARY DATASETS
//*****
//BR14 EXEC PGM=IEFBR14
//SYSLIN DD DISP=(,PASS,DELETE),DSN=&&SYSLIN,
//          UNIT=SYSALLDA,SPACE=(CYL,(1,3,5)),
//          DCB=(LRECL=80,BLKSIZE=27920,DSORG=PO,RECFM=FB)
//SYSLMOD DD DISP=(NEW,CATLG),DSN=&TMPDBD,
//          UNIT=SYSALLDA,SPACE=(CYL,(1,3,5)),
//          DCB=(&DBDLIB,DSORG=PO)
//*****
//*        ASSEMBLE SOURCE MEMBER
//*****
//ASM1 EXEC PGM=ASMA90,COND=(5,LE),
//          PARM='OBJECT,NODECK,NOUSING,SIZE(MAX,ABOVE)'
//SYSLIB DD DISP=SHR,DSN=&IMSMAC
//          DD DISP=SHR,DSN=SYS1.MACLIB
//          DD DISP=SHR,DSN=SYS1.MODGEN
//SYSLIN DD DISP=(OLD,PASS),DSN=&&SYSLIN(&MBR.)
//SYSTEM DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=VIO,DISP=(,DELETE),SPACE=(CYL,(10,5))
//SYSIN DD DISP=SHR,DSN=&SRCLIB(&MBR.)
//*****
//*        LINK CONTROL BLOCK
//*****
//LNK1 EXEC PGM=IEWL,COND=(5,LE),
//          PARM='LIST,XREF,MAP,LET,REUS,NCAL'
//SYSLMOD DD DISP=OLD,DSN=&TMPDBD(&MBR.)
//SYSLIN DD DISP=(OLD,PASS),DSN=&&SYSLIN(&MBR.)
//SYSUT1 DD UNIT=VIO,SPACE=(TRK,(10,10))
//SYSPRINT DD SYSOUT=*
//*****
//*        ONLINE REORG
//*****
//D2REORG EXEC PGM=NSOMAIN,REGION=0M,DYNAMNBR=250
//STEPLIB DD DISP=SHR,DSN=&NEONLIB
//          DD DISP=SHR,DSN=&RESLIB
//          DD DISP=SHR,DSN=&EXITLIB
//IMSDALIB DD DISP=SHR,DSN=&MDALIB
//          DD DISP=SHR,DSN=&RESLIB
//IMS DD DISP=SHR,DSN=&PSBLIB
//          DD DISP=SHR,DSN=&DBDLIB
//NEWIMS DD DISP=SHR,DSN=&PSBLIB
//          DD DISP=(OLD,DELETE),DSN=&TMPDBD
//          DD DISP=SHR,DSN=&DBDLIB

```

Now execute the previous JCL proc.

```
//          <<< Your Job Card Goes Here >>>
//OLREORG EXEC NEOND2,
//          MBR=DDAHODB,           Database being reorganized
//          PSBLIB='IMS.PSBLIB',   PSB Library
//          DBDLIB='IMS.DBDLIB',   DBD Library
//          SRCLIB='IMS.DBDSRC',   Module Source Library
//          MDALIB='IMS.MDALIB',   IMS Dynamic Alloc. Library
//          RESLIB='IMS.RESLIB',   IMS Reslib
//          NEONLIB='NEON.LOAD',   NEON Product Library
//          TMPDBD='IMS.TEMP.DBDLIB', Temporary DBD Library
//          EXITLIB='IMS.USEREXIT'  COMPRESSION EXIT LIBRARY
//D2REORG.NSOSYSIN DD *
//          REORG DBD(DDAHODB) -
//          MODE(ONLINE)
//
```

Database Director will stop processing and put out a WTOR that must be responded to:

```
NSO2200I Changes were detected in the following database definitions:
          DDAHODB
NSO2100I Database Director waiting for IMS Online Change
          Reply "FINISH" or "CANCEL"
```

At this point, you are ready to perform your ACB gen and online change. After you have successfully issued your /MOD PREPARE ACBLIB and /MOD COMMIT, you can reply "FINISH" to the Database Director outstanding reply.

# Summary

Here is a summary of the basic steps for implementing hardware data compression:

1. Create an unload file of each database segment type.

Using the Database Director Cloning feature and Eclipse iExtract, build multiple unload files. Each file contains a specific segment type. This is accomplished in a single job step using online production data and without any online IMS outages.

2. Build the hardware data compression exit routines.

Using the unload files from the previous step, create segment compression exit routines that are unique to each database segment.

3. Modify the DBD.

Add a COMPRTN= parameter to each segment definition within the DBD. Ensure the following:

- a. If there is an existing segment compression exit routine, build a new compression exit routine with a different name. This allows Database Director to recognize that there is a compression exit change.
  - b. Specify "INIT" as the third parameter on the COMPRTN parameter. Hardware data compression requires this setting.
4. Perform the online reorganization using Database Director.

This is a "special" Database Director online reorganization job because the generated DBD is in a temporary DBD library that is used in the online reorganization step.

5. Perform a DBD gen, ACB gen and online change.

When the Database Director online reorganization is ready to restart the database, it issues a WTOR indicating that it is waiting for you to complete the online change.

6. Finally, issue the "FINISH" reply to the Database Director outstanding reply. This allows the online reorganization to restart the database and resume any suspended DLI, DBB and/or BMP jobs.

## Shortening the Process

With variations, you may be able to shorten the database outage even further. For example, the following steps can be accomplished while the Database Director online reorganization is running:

1. Perform a “temporary” DBD gen
2. Using the “temporary” DBD library, perform an ACB gen
3. Run the Online Change Utility

After Database Director issues its WTOR, simply issue and verify the IMS MODIFY commands. Don't forget to generate the DBD into a “real” DBD library.

## Versatility and Availability

Database Director and Eclipse iExtract can improve online database availability and much more. Using patented technology, these products provide high-speed IMS data extract functionality while maintaining online IMS application availability.

Eclipse iExtract is a powerful data extract tool that provides high-speed access to IMS databases. Database Director is an online database reorganization and cloning tool that maintains online IMS application availability and data integrity. Combined, they provide a solution to any number of data availability issues – both online and batch.

NEON Enterprise Software delivers innovative products that enhance your business objectives.