



Using Multiple BCV Database Copies with IBM[®] DB2 Universal Database[®] V7.2 and EMC TimeFinder[®]

April 2, 2002

EMC PARTNUM: (H535)

IBM (TR-74.186)

John Macdonald
EMC Corporation

Enzo Cialini
IBM Canada Ltd.
IBM Toronto Lab

Copyright © 1999, 2000, 2001, 2002 EMC Corporation and IBM Corporation. All rights reserved.

EMC and IBM believe the information in this publication is accurate as of its publication date. The information is subject to change without notice.

THE INFORMATION IN THIS PUBLICATION IS PROVIDED “AS IS.” EMC CORPORATION AND IBM CORPORATION MAKE NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WITH RESPECT TO THE INFORMATION IN THIS PUBLICATION, AND SPECIFICALLY DISCLAIM IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Use, copying, and distribution of any EMC or IBM software described in this publication requires an applicable software license.

The furnishing of this document does not imply giving license to any IBM or EMC patents.

References in this document to IBM products, Programs, or Services do not imply that IBM intends to make these available in all countries in which IBM operates.

EMC, EMC², and Symmetrix are registered trademarks and Engenuity, TimeFinder, and where information lives are trademarks of EMC Corporation.

IBM, AIX, DB2, DB2 Universal Database, and RS/6000 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

All other brand names are trademarks or registered trademarks of their respective owners.

EMC Part Number: H535; IBM Part Number: TR-74.186

Printed 4/17/2002

Table of Contents

Introduction	4
TimeFinder Business Continuance Sequence	4
Relocating a Copy	5
Uses of a Relocated Copy	5
AIX recreatevg Command	5
DB2 UDB db2relocatedb Command	5
DB2 UDB db2inidb Command	6
Test Configuration	7
Configuration Setup Tasks.....	10
Document Command Conventions	10
Initial Device Setup Tasks (Both Systems).....	10
Setting Up Subsequent Symmetrix Device Groups for TimeFinder Operations.....	11
Configuring STD Data on the AIX Systems.....	13
Using the First BCV Copy on the Primary System	15
Split the BCV Copy	15
Duplicating the Disk Configuration	16
Preparing the Relocation Configuration File	17
Relocating the Database.....	18
Refreshing a Copied Database.....	20
Terminating DB2 Use of a Database Copy	20
Removing a Database Copy: Drop from AIX.....	21
Refreshing the Disk Subsystem	21
Reinitializing the Refreshed Copy for Use.....	22
Using a Second Database Copy on the Primary System	23
Using Multiple Database Copies on a Secondary System.....	24
Using a Relocated Database Copy on the Secondary System.....	25
Appendix A: Procedures.....	26
Symmetrix Command Access.....	26
AIX and Symmetrix Volume Discovery	26
Setting Up Symmetrix Device Groups for TimeFinder Operations.....	26
Establishing an Initial BCV Mirror	27
Splitting a Nonbusy BCV Group.....	27
Splitting a Busy BCV Group.....	28
Setting Up an Additional BCV Device Group for TimeFinder Operations	28
Creating a Volume Group Containing a List of Volumes	29
Creating a Logical Volume for Journalled File System.....	29
Creating a Journalled File System.....	30
Creating a Logical Volume to Contain Raw Disks	30
Creating a DB2 Database in a Journalled File System.....	31

Configuring a Raw Device as DMS in a Database	31
Setting the Location of Logs	32
Importing a Nonrelocated Disk Configuration to a Secondary System.....	32
Cataloging a Nonrelocated Database on the Secondary System	32
Duplicating the Disk Configuration to use an Additional BCV Copy	33
Database Relocation File for an Additional BCV Copy of a Database	34
Relocating an Additional BCV Copy of a Database	35
Preparing for Reestablish on the Secondary System.....	36
Reestablishing on the Primary System.....	37
Removing a Database Copy: Drop from AIX.....	37
Updating the Database BCV Copy from the Standard	38

Introduction

Mission-critical database systems must operate 24x7 with the highest degree of availability possible. As databases increase in size and ad hoc queries place more demands on the continued availability of the system, the time and hardware resources required to back up and recover databases grows substantially while the maintenance windows are either drastically reduced or disappear completely. The instant split feature of EMC TimeFinder™ software and the Database Suspend I/O features available in IBM® DB2® Universal Database™ (UDB) combine to provide highly available, advanced database technical architectures to meet these demands. This paper will demonstrate the use of these features to create a coherent copy of a continuously available DB2 UDB database to meet the increasing availability requirements demanded in today's business environment.

An earlier paper (IBM Tech Report TR-74.180; EMC White Paper JM52094, **Creating Hot Snapshots and Standby Databases with IBM DB2 Universal Database V7.2 and EMC TimeFinder**) demonstrated a sample configuration setup and usage of IBM DB2 UDB Enterprise Edition (EE) V7.2 with EMC TimeFinder that allowed a primary system to provide ongoing access to the database, while a copy of the database was made available to a secondary system for auxiliary purposes.

This paper extends the paper noted above to demonstrate the handling required to use multiple copies of a database on the same system – either allowing secondary processing to occur on the primary system (without pausing the main application) or allowing multiple concurrent copies to be used on a secondary system for separate purposes. This paper limits duplication of the material in the previous paper; there is much less background exposition provided for those steps that are unchanged from that paper. The procedures that are being reused are recapped in Appendix A. There is less commentary detail than was given in the previous paper, and in some cases, the procedures are modified to be less specific to the scenarios that were examined in that white paper.

This paper also makes use of the `db_split` script that can be downloaded from:

ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest

TimeFinder Business Continuance Sequence

With EMC TimeFinder, a Symmetrix® standard device can be paired with one or more business continuance volume (BCV) devices. A BCV is an independently host-addressable device that is a point-in-time copy of its paired standard device. A BCV device can be assigned to the same host as the standard device, or to a different host.

A standard device and a BCV can be in a number of states. Initially, the BCV is established to the standard device, making a BCV pair. TimeFinder copies all tracks from the standard device to the BCV. While in the establishing/established state, the BCV becomes unavailable from its original host address. Once established, the BCV copy can be split from the standard device, which again makes it available to the host, for other tasks. An incremental establish resynchronizes a previously split BCV/standard device pair. For an incremental establish, TimeFinder copies only updated tracks from the standard to the BCV device, and refreshes any BCV tracks that were changed. During this process, the BCV is not available at its original host address. Similarly, the restore and incremental restore commands copy all tracks, and only changed tracks, respectively, from the BCV to the standard device, during which process the BCV is not available at its original host address.

Relocating a Copy

A database has several characteristics that must be unique and some that can be in common with other databases. The combination of database name, instance name, and base path must be unique on a system. Directories used for logs and paths used for containers must be used for only one database.

For multiple copies of a database to be available on one system, these attributes must be changed for all of the copies – only one can have the original attributes. On the original system, that is of course the original database.

On a secondary system, one copy can be available with the original attributes. Any other copies on either system must be relocated. AIX® and DB2 provide commands that are useful for this relocation process.

These attributes are used by DB2 — it has internal information about the location of each database and instance, and so on. So, it is not sufficient to simply mount the copies in a different location — DB2 must also be informed about the changes.

Uses of a Relocated Copy

A database copy that has been relocated can be used for the same purposes as an unrelocated copy. It is particularly useful for tasks that will not affect the original database: testing new application code (where you do not want the test operations applied to the real database), or reporting (where you do not want database updates to make the data collected for the initial stages of the report inconsistent with the data collected for later stages). It is somewhat less useful for tasks that might affect the real database: backups and standby. In those cases, the relocation would have to be undone when the copy was being used to fix the original database. It is better to use an unrelocated copy for these tasks.

AIX recreatevg Command

The command `recreatevg [-y VGNAME] [-Y LVPREFIX] PVNAME ...` gives the specified physical volumes new volume *Ids*, and recreates the volume group, logical volume, and file systems that were on them with revised names. This is important to prevent AIX from confusing different copies of a volume.

DB2 UDB db2relocatedb Command

The command `db2relocatedb -f CONFIG_FILE` allows the use of a copy of a database. The configuration file contains lines that specify the old and new value of aspects that have changed. This can include the database name, the path to the database, the instance name, the node number, the log directory, and container paths. This command was designed to relocate an existing database. Thus, if executed within the same instance it will have the effect of moving the existing database. For the purposes of this paper, a different instance will always be used when making a relocated copy, either on the original system or on a secondary system.

For further information on the `db2relocatedb` command and the structure/values of the `CONFIG_FILE`, refer to the DB2 UDB Documentation and Release Notes provided with DB2 UDB V7 FP4. They are available from:

<ftp://ftp.software.ibm.com/ps/products/db2/info/vr7/pdf/letter/db2ire72.pdf>

DB2 UDB *db2inidb* Command

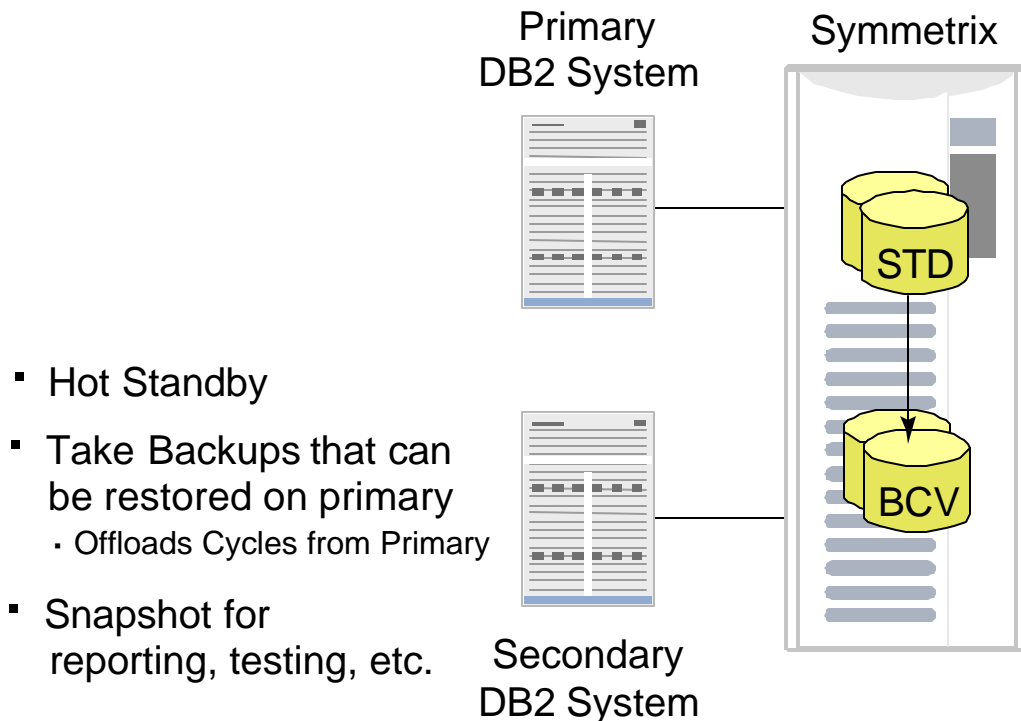
The *db2inidb* command *db2inidb <db_alias> as < snapshot | standby | mirror > [relocate using <configfile>]* is required to initialize the copy of the suspended database. It can be used in the following three cases:

- *snapshot* can be applied to the secondary copy, putting it into a transactionally consistent state.
- *standby* can be applied to the secondary copy, putting it into a rollforward pending state. DB2 logs from the primary database can then be applied to the secondary database.
- *mirror* can be applied to the primary copy after it has been restored from the secondary copy. The primary database will be placed into a rollforward pending state, and then DB2 logs can be applied to the primary database.

The *relocate using* option combines the effect of the *db2relocatedb* command into the initialization.

Test Configuration

These procedures were tested using an EMC Symmetrix 8430 running Symmetrix Enginuity™ Operating Environment version 5566. Two IBM RS/6000® H70 systems were connected via fiber HBA to the Symmetrix system. One RS/6000 system was used as a primary database system. The second RS/6000 system was used as a secondary database system (either as a standby system, or as a testing/analysis system). The following figure illustrates the test configuration.



- Hot Standby
- Take Backups that can be restored on primary
 - Offloads Cycles from Primary
- Snapshot for reporting, testing, etc.

Figure 1. DB2 UDB and EMC Test Configuration

The two RS/6000 systems were running AIX 4.3.3 and IBM DB2 UDB Enterprise Edition, V7.2 FP4.

The interaction between the RS/6000 systems and the Symmetrix system was done with AIX BCV support kit 2.0, and the EMC AIX kit 4.3.3.2, both from EMC. The set of program temporary fixes (PTF) listed in the EMC Open Source Matrix as of March 2001 was applied.

The primary system storage for the database was contained on twenty-four 4 GB Symmetrix hypervolumes (or hypers) with Symmetrix device numbers 000 to 017. The corresponding BCV *mirrors* of this storage were configured on other groups of twenty-four 4 GB Symmetrix hypers. Device numbers 080 to 097 and 098 to 0AF were used on the primary system, and device numbers 0C0 to 0D7 and 0D8 to 0EF were used on the secondary system. Gatekeeper devices are used by SYMCLI commands to communicate with the Symmetrix system. This configuration had four small gatekeeper volumes for each RS/6000 system.

Primary storage devices are also referred to as Standard (STD) devices on the Symmetrix system. The 24 STD hypers were configured into two volume groups on the RS/6000 systems. The first volume group, *drv*, consisted of 20 hypers, and was further divided into two logical volumes. The first logical volume, *dlv*, consisted of four hypers, which were used as a journalled file system containing the base data for our database. The other 16 hypers in this volume group were configured into a logical volume, *rlv*, which was used as a raw logical volume for the majority of the data in the database. The final four hypers were configured into a second volume group containing one logical volume, *llv*, used as a journalled file system to contain the logs for the database.

The following tables show the Symmetrix device group name, AIX volume group name, AIX logical volume name, file system mount points (device name in the case of raw logical volumes), the Symmetrix device numbers, and the AIX hdisk names assigned to the Symmetrix hypervolumes used for each database copy.

Table 1. Primary System – Standard Volumes – testdb

Volume Group	Logical Volume	Mount Point	Standard Symmetrix dev	hdisk
drv	dlv	/data	000:003	7:10
drv	rlv	/dev/rrlv	004:013	11:26
lv	llv	/logs	014:017	27:30

Table 2. Primary System – First BCV Copy – test1db

Volume Group	Logical Volume	Mount Point	BCV Symmetrix dev	hdisk
dr1vg	c1dlv	/copy1/data	080:083	31:34
dr1vg	c1rlv	/dev/rc1rlv	084:093	35:50
l1vg	c1llv	/copy1/logs	094:097	51:54

Table 3. Primary System – Second BCV Copy – test2db

Volume Group	Logical Volume	Mount Point	BCV Symmetrix dev	hdisk
dr2vg	c2dlv	/copy2/data	098:09B	55:58
dr2vg	c2rlv	/dev/rc2rlv	09C:0AB	59:74
l2vg	c2llv	/copy2/logs	0AC:0AF	75:78

Table 4. Secondary System – First BCV Copy (In Original Position) – testdb

Volume Group	Logical Volume	Mount Point	Standard Symmetrix dev	hdisk
drv	dlv	/data	0C0:0C3	7:10
drv	rlv	/dev/rrlv	0C4:0D3	11:26
lv	llv	/logs	0D4:0D7	27:30

Table 5. Secondary System – Second BCV Copy (In Relocated Position) – test1db

Volume Group	Logical Volume	Mount Point	BCV Symmetrix dev	hdisk
dr1vg	c1dlv	/copy1/data	0D8:0DB	31:34
dr1vg	c1rlv	/dev/rc1rlv	0DC:0EB	35:50
llvg	c1llv	/copy1/logs	0EC:0EF	51:54

The following were demonstrated using this configuration:

- Creating two copies of a live production database on the production system, in addition to retaining the original live production database.
- Creating two copies of a live production system on a secondary system with one of them in the same location as the original, the other a relocated copy.

Configuration Setup Tasks

A fair amount of planning and configuration goes into setting up the storage for a large database implementation on an EMC Symmetrix system. Before you can use Symmetrix standard or TimeFinder volumes, the physical drives in the Symmetrix system must be logically subdivided into hypervolumes, which can appear to your system as physical disks. Symmetrix hypervolumes are further configured as STD, BCV, and other Symmetrix device types. Once this is complete, the devices must be made known to your operating system. This step is operating system (OS) specific; in this case, the OS is AIX 4.3.3. The same procedures could be done on other OSs, such as HP/UX or Solaris™, but the commands to manage the devices would need to be changed into the appropriate OS-specific equivalents.

The EMC Solutions Enabler SYMCLI Base Component software version 4.2 is required for the SYMCLI functionality described in this document. SYMCLI commands must be installed and available to the root user doing the initial device configuration and subsequent operational steps. Finally, the SYMCLI database is initialized to include the new devices so that later SYMCLI commands can operate efficiently on those devices.

This document will not provide tutorial details about the basic aspects of configuration that were already covered in detail in the companion white paper.

Document Command Conventions

Commands that must be run by root will be shown with a shell prompt of `root>`.

Commands that must be run by the original instance owner of the database will be shown with a shell prompt of `dba>`.

Commands that must be run by a different database instance from the original database will be shown with a shell prompt of `altdba>`.

Commands in boldface are shown as the actual commands issued on one (or both) of the systems.

Initial Device Setup Tasks (Both Systems)

This setup is largely the same as was done for the previous white paper. See the other paper for details about what steps are being done here.

```
root> # Symmetrix Command Access
root> export PATH=$PATH:/usr/symcli/bin
root> export PATH=$PATH:/usr/lpp/Symmetrix/bin
root> # AIX and Symmetrix Volume Discovery
root> emc_cfgmgr
root> symcfg discover
root> symdev list

root> # Set Up Symmetrix Device Groups for TimeFinder Operations
root> symdg -type REGULAR create pr1
root> symld -g pr1 -RANGE 000:017 addall dev
root> symbcv -g pr1 -RANGE 080:097 associateall dev

root> # Establish an Initial BCV Mirror
root> symmir -g pr1 -noprompt -full -exact establish

root> # Split a Non-Busy BCV Group
root> symmir -g pr1 -I 30 verify
root> symmir -g pr1 -noprompt -instant split
```

Setting Up Subsequent Symmetrix Device Groups for TimeFinder Operations

Rather than setting up a separate Symmetrix device group for each of data and logs (as was done in the previous white paper) a single device group containing both of them is used. None of the operations considered in this white paper require separate treatment for the logs.

Additionally, we have multiple BCV copies to manage, so having one group per BCV instead of two makes the management easier. It is straightforward to revert to using multiple device groups if your needs are different.

Although there are multiple BCV copies, they are all copies of the same standard device set. Creating the device group for the first BCV is the same as in the previous white paper, and it is shown in the previous section without comment. However, for the subsequent BCV copies, the standard devices must be moved into the new group from the group they were in previously.

General Procedure

Values required to run procedure:

OGROUP	Name for the Symmetrix device group currently containing the STD Symmetrix volumes to be mirrored with BCV volumes.
NGROUP	Name for the new Symmetrix device group to also be mirrored with STD volumes.
BSTART	First BCV volume (three-digit hex Symmetrix device number).
BEND	Last BCV volume (three-digit hex Symmetrix device number).

1. Create the group for the new BCV volumes:

```
root> symdbg -type REGULAR create NGROUP
```
2. Move the STD primary devices to the new group:

```
root> symld -g OGROUP moveall NGROUP
```
3. Associate a range of BCV devices with the group:

```
root> symbcv -g NGROUP -RANGE BSTART:BEND associateall dev
```
4. Repeat step 3 if you want to combine nonconsecutive blocks of volumes into a single BCV group. Then, establish the BCV, wait for the establish to complete, and split it.

```
root> symmir -g NGROUP -noprompt -full -exact establish
root> symmir -g NGROUP -I 30 verify
root> symmir -g NGROUP -noprompt -instant split
```

Performed on Both Systems

```
root> # Set Up additional BCV Device Groups
root> symdbg -type REGULAR create pr2
root> symld -g pr1 moveall pr2
root> symbcv -g pr2 -RANGE 098:0AF associateall dev
root> symmir -g pr2 -I 30 verify
root> symmir -g pr2 -noprompt -instant split
root> #
root> symdbg -type REGULAR create sc1
root> symld -g pr2 moveall sc1
root> symbcv -g sc1 -RANGE 0C0:0D7 associateall dev
root> symmir -g sc1 -I 30 verify
root> symmir -g sc1 -noprompt -instant split
root> #
root> symdbg -type REGULAR create sc2
root> symld -g sc1 moveall sc2
root> symbcv -g sc2 -RANGE 0D8:0EF associateall dev
root> symbcv -g sc2 -RANGE 098:0AF associateall dev
root> symmir -g sc2 -I 30 verify
root> symmir -g sc2 -noprompt -instant split
```

Configuring STD Data on the AIX Systems

Shell variables were created to group the hdisk volumes according to their uses. (See Tables 1 to 5 above in the Test Configuration.) Not all of these variables will be used, and some are only used for specific procedures that might not be required for your purposes. However, it is easier to define them all than to trace through and figure out which ones will actually be used and define only those, especially if your usage plans change and you later have to go back and add some definitions. The variables use the name prefixes **data**, **raw**, and **logs** for the groups of disks used for those purposes, and the prefix **dr** for the combination of **data** and **raw** (which together contain the database in our configuration.)

Then you can use those variables to prepare the volumes for use and to setup the database. Note that you might have to write these separately for the primary and secondary system unless you have the same hdisk volume names on both systems (as in the test setup).

Performed on the Primary System

```
root> symmir -g sc2 moveall pr1
root> symmir -g pr1 -noprompt establish
```

Performed on Both Systems

Standard volumes on primary, unrelocated BCV copy on secondary:

```
root> datadk='hdisk7 hdisk8 hdisk9 hdisk10'
root> rawdk='hdisk11 hdisk12 hdisk13 hdisk14'
root> rawdk="$rawdk hdisk15 hdisk16 hdisk17 hdisk18"
root> rawdk="$rawdk hdisk19 hdisk20 hdisk21 hdisk22"
root> rawdk="$rawdk hdisk23 hdisk24 hdisk25 hdisk26"
root> logsdk='hdisk27 hdisk28 hdisk29 hdisk30'
root> drdk="$datadk $rawdk"
```

First relocated BCV copy on primary and secondary:

```
root> data1dk='hdisk31 hdisk32 hdisk33 hdisk34'
root> raw1dk='hdisk35 hdisk36 hdisk37 hdisk38'
root> raw1dk="$raw1dk hdisk39 hdisk40 hdisk41 hdisk42"
root> raw1dk="$raw1dk hdisk43 hdisk44 hdisk45 hdisk46"
root> raw1dk="$raw1dk hdisk47 hdisk48 hdisk49 hdisk50"
root> logs1dk='hdisk51 hdisk52 hdisk53 hdisk54'
root> dr1dk="$data1dk $raw1dk"
```

Performed on the Primary System

Second relocated BCV copy on primary system (only one relocated copy was used on secondary for these tests):

```
root> data2dk='hdisk55 hdisk56 hdisk57 hdisk58'
root> raw2dk='hdisk59 hdisk60 hdisk61 hdisk62'
root> raw2dk="$raw2dk hdisk63 hdisk64 hdisk65 hdisk66"
root> raw2dk="$raw2dk hdisk67 hdisk68 hdisk69 hdisk70"
root> raw2dk="$raw2dk hdisk71 hdisk72 hdisk73 hdisk74"
root> logs2dk='hdisk75 hdisk76 hdisk77 hdisk78'
root> dr2dk="$data2dk $raw2dk"
```

```
root> # create volume groups
root> mkvg -f -ydrv -s64 $datadk $rawdck
root> mkvg -f -ylvg -s64 $logsdck

root> # create logical volumes
root> mklv -ydlv -tjfs -x250 drvg 1 $datadk
root> mklv -yllv -tjfs -x250 lvg 1 $logsdck
root> mklv -yrlv -x1000 drvg 1000 $rawdck

root> # Create journalled file systems
root> crfs -vjfs -ddl -m/data -Ayes -prw -tno -a nbpi=8192
root> crfs -vjfs -dllv -m/logs -Ayes -prw -tno -a nbpi=8192
root> chfs -a size=32768000 /data
root> chfs -a size=32768000 /logs
root> mount /data
root> mount /logs

root> # Make the space available to the database administrator
root> chown dba /data /logs /dev/rrlv

dba> # Start the DB2 UDB instance, if not already started
dba> db2start

dba> # Create the database on the journalled file system using
dba> # default SMS Containers for the Catalog, User, and Temporary Tablespaces.
dba> db2 create db testdb on /data

dba> # Create the DMS Tablespace using the raw volume/device
dba> db2 connect to testdb
dba> db2 "create tablespace testraw managed by database \
        using ( device '/dev/rrlv' 60G )"

dba> # Move the DB2 LOG files from their default location to the created log
dba> # filesystem.
dba> db2 update db cfg for testdb using NEWLOGPATH /logs
```

Using the First BCV Copy on the Primary System

We now show how to use another copy of the database on the primary system. Since the original database is still present, we must use a different DB2 instance (account) for the copy. Otherwise, DB2 will update the information about the original database instead of creating a new database with the copy. While this copy is being taken and made ready for use, the original database can continue normal operations without any visible interruption. The steps for making the copy are:

- Split the BCV copy
- Use `recreatevg` to duplicate the volume layout (with different names)
- Prepare a configuration file
- Relocate the copied database

Split the BCV Copy

The first BCV copy has been establishing. It should be established and thus ready to split off and use.

Splitting the BCV is the same procedure as in the previous white paper:

```
root> symmir -g pr1 -i 30 verify  
root> db_split dba testdb pr1
```

The script `db_split` can be downloaded from:

```
ftp://ftp.emc.com/pub/symm3000/DB2/db\_split\_latest
```

Duplicating the Disk Configuration

The volume manager setup, file system setup, and database setup have already been done on the STD volumes and copied to the BCV volumes. Next, you need to create a layout that has the same components, but names them differently so that AIX does not confuse them with the components of the original copy.

General Procedure

Values required to run procedure:

VGNAME	Name of the volume group being created.
DISKS	Device names of the disks in the group.
DIRPRE	The directory prefix to be used to contain the mount points of the copy.
DEVPRE	The prefix to be used to modify the device names from their original values.
ALTDBA	The instance name of the new database copy (which must be different from the instance name of the original database).
DATA	The mount point of the original database data.
LOGS	The mount point of the original database logs.
RAWDEV	The device name of the original raw device (omitting the leading letter "r").

1. Create a new volume group definition from the physical volumes that is consistent with the original layout:

```
root> recreatevg -yVGNAME -YDEVPRE -LDIRPRE DISKS
root> mount DIRPRE/DATA
root> mount DIRPRE/LOGS
```

Note: The mount points of the newly created volume group are placed under the provided prefix but are otherwise the same. This illustrates one way of maintaining a consistent naming layout; but the mount points can be anywhere. It is very useful to create a clear naming strategy to organize multiple copies of a database.

2. The files and raw device, must be given to the new instance.

```
root> chown -R ALTDBA DIRPRE/DATA DIRPRE/LOGS rDEVPRERAWDEV
```

Performed on the Primary System

```
root> recreatevg -ydr1vg -Yc1 -L/copy1 $dr1dk
root> recreatevg -y11vg -Yc1 -L/copy1 $11dk $logs1dk
root> mount /copy1/data
root> mount /copy1/logs
root> chown -R altdba /dev/rc1rlv /copy1/data /copy1/logs
```

Preparing the Relocation Configuration File

The database is already present on the imported disk volumes, but IBM DB2 UDB has not yet been made aware of these imported volumes. Furthermore, the internal information within the copied data still refers to the positioning of the original database and not the relocated positioning of the copy. A preliminary step to informing IBM DB2 UDB of the imported volumes is to create a configuration file that specifies the changes that have been made from the original to create the copy.

General Procedure

Values required to run procedure:

ODBNAM	Original name of the database.
NDBNAM	Name of the new copy.
OINST	Original instance name.
NINST	New instance owner name.
OPATH	Location of the original database.
NPATH	Location of the new copy.
OLOGS	Location of the original logs.
NLOGS	Location of the new logs.
OCONT	Location of an original container.
NCONT	Location of the copied container.

Create a configuration file with the relocation information:

```
DB_NAME=ODBNAM , NDBNAM
INSTANCE=OINST , NINST
DB_PATH=OPATH , NPATH
LOG_DIR=OLOGS , NLOGS
CONT_PATH=OCONT , NCONT
```

The CONT_PATH entry can be repeated if your database has multiple containers, it can be omitted if your database has no containers or if they are all below the database path (i.e. DB_PATH). The LOG_DIR entry can also be omitted if the logs are kept under the DB_PATH directory. There can also be an entry for NODENUM that can be used in an EEE environment.

Performed on the Primary System

```
# The file was created in copy1.conf
altdba> cat copy1.conf
DB_NAME=testdb,testldb
INSTANCE=dba,altdba
DB_PATH=/data,/copy1/data
LOG_DIR=/logs,/copy1/logs
CONT_PATH=/dev/r1rlv,/dev/rc1rlv
altdba>
```

Note: The copy1.conf file must be accessible from the system on which the db2relocatedb or db2inidb commands are executed.

Relocating the Database

The IBM DB2 UDB relocate function will apply the configuration changes to the internal fields of the database, but there is one external aspect that must first be corrected. The pathname used for the database contains the instance name as one component. The copied BCV still has the old instance name in that path. It must be changed and then it is simply a matter of starting up DB2 and informing it about the new data and the relocation that must be applied to it.

General Procedure

Values required to run procedure:

OINST	Original instance name.
NINST	New instance owner name.
NPATH	Location of the new database copy.
NDBNAM	Name of the new database.
CONF	Relocation configuration file.

1. Put the database into the right path for the instance:

```
root> mv NPATH/OINST NPATH/NINST
```

2. Tell DB2 about the new database and have it relocate the internal information:

```
altdba> db2start # (if it is not already started)
altdba> db2inidb NDBNAM as snapshot relocate using CONF
```

Performed on the Primary System

```
root> mv /copy1/data/dba /copy1/data/altdba

altdba> db2start
altdba> db2inidb test1db as snapshot relocate using copy1.conf
```

The copied database is now available for use by the new DB2 instance.

Refreshing a Copied Database

A copied database can be used for many purposes. Such purposes often must be repeated on a fresh copy of the database. For example, a monthly reporting process that uses a copy of the database so that no changes will be applied while the report is being collected is repeated each month on a fresh copy of the database.

Refreshing the copy to synchronize it with the latest data from the primary database is essentially a matter of:

- Terminating any use of the old copy
- Dropping the volumes from the system
- Reestablishing the BCV to update its content
- Splitting and reinitialize the copy

Terminating DB2 Use of a Database Copy

Before AIX will permit the volumes to be released, there must be no outstanding activities connected to them.

General Procedure

Value required to run procedure:

NDBNAM	Name of the new database.
---------------	---------------------------

If this is the only DB2 database running under the instance on that system, tell DB2 to stop using the database copy that is about to be overwritten:

```
altdba> db2stop force
```

Alternately, if there are other databases running under the instance that you do not want to interrupt, you must first determine which applications are accessing the database that is about to be refreshed, force them off, and then uncatalog the database to prevent future connections:

```
altdba> db2 list applications
```

Find the handle of each application that must be terminated (say H1, H2, etc.) and then:

```
altdba> db2 'force application ( H1[,H2]... )'
```

Once all applications have been forced, uncatalog the database:

```
altdba> db2 uncatalog database NDBNAME
altdba> db2 terminate
```

Performed on the Primary System

```
altdba> db2stop force
```

Removing a Database Copy: Drop from AIX

General Procedure

Values required to run procedure:

DIRPRE	Prefix directory for copied data and logs.
DBPATH	Path to the original database.
LOGS	Path to the original logs.
DVGNAM	New data volume group name.
LVGNAM	New logs volume group name.

Make AIX forget about the database copy that is about to be overwritten:

```
root> umount /DIRPRE/DBPATH
root> umount /DIRPRE/LOGS
root> varyoffvg DVGNAM
root> varyoffvg LVGNAM
root> exportvg DVGNAM
root> exportvg LVGNAM
```

Performed on the Primary System

```
root> umount /copy1/data
root> umount /copy1/logs
root> varyoffvg dr1vg
root> varyoffvg l1vg
root> exportvg dr1vg
root> exportvg l1vg
```

Refreshing the Disk Subsystem

The BCV was established to refresh it to the current content of the primary database. Since a single device group was used to contain all of the data and logs volumes, the group version of the `db_split` command can be used instead of creating a device list file and using the file version. Otherwise, this is the same procedure that is done to reestablish a BCV copy used with relocation on a secondary system.

Performed on the Primary System

```
root> # Establish the BCVs
root> symmir -g pr1 -noprompt establish
root> symmir -g pr1 -i 30 verify

root> # Split the BCVs
root> db_split -g testdb dba pr1
```

The script `db_split` can be downloaded from:

```
ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest
```

Reinitializing the Refreshed Copy for Use

Initializing the copy for use after it has been reestablished is the same as the initialization done earlier after it was first created.

Performed on the Primary System

```
root> recreatevg -ydr1vg -Yc1 -L/copy1 $dr1dk
root> recreatevg -y11vg -Yc1 -L/copy1 $11dk
root> mount /copy1/data
root> mount /copy1/logs

root> chown -R altdba /dev/rc1rlv /copy1/data /copy1/logs
root> mv /copy1/data/dba /copy1/data/altdba

altdba> db2start
altdba> db2inidb test1db as snapshot relocate using copy1.conf
```

The database copy is again ready for use, but has been updated to a current copy of the primary database.

Using a Second Database Copy on the Primary System

You might want to use two copies of the database at the same time, perhaps for different purposes. A second copy can be established and used exactly the same way as the first. While the second copy cannot use the same instance user as the original database, it can use the same instance user as another copy. (It does not have to use the same instance though.) While this second copy is being created, the original and the first copy can be in use.

Performed on the Primary System

```
root> # assumes STD are in group pr1 and already split
root> symld -g pr1 moveall pr2
root> symmir -g pr2 -noprompt establish
root> symmir -g pr2 -i 30 verify
root> db_split dba testdb pr2

root> recreatevg -ydr2vg -Yc2 -L/copy2 $dr2dk
root> recreatevg -yl2vg -Yc2 -L/copy2 $l2dk
root> mount /copy2/data
root> mount /copy2/logs
root> chown -R altdba /dev/rc2rlv /copy2/data /copy2/logs

altdba> cat copy2.conf
DB_NAME=testdb,test2db
INSTANCE=dba,altdba
DB_PATH=/data,/copy2/data
LOG_DIR=/logs,/copy2/logs
CONT_PATH=/dev/rmlv,/dev/rc2rlv
altdba>

root> mv /copy2/data/testdbaltdba /copy2/data/test2dbaltdba

altdba> db2start
altdba> db2inidb test2db as snapshot relocate using copy2.conf
```

The second copy is now available for use. The script `db_split` can be downloaded from:

```
ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest
```

Using Multiple Database Copies on a Secondary System

Using copies of a database on a secondary system is also a straightforward combination of what has already been shown. One copy can be used in the original location, as was shown in the previous white paper. Multiple copies can be used in relocated locations. If there is a nonrelocated copy on a system or more than one relocated copy, the relocated copies may not be in the same instance as the original.

A typical scenario might be to have one nonrelocated BCV copy that is generally left in the established state, ready to be split and used to take over responsibility for the database application in the case of a system crash on the primary system. Occasionally, this BCV could also be split to take a backup.

Meanwhile, one or more relocated BCV copies could be used for other purposes, such as generating periodical reports.

Below, you can see how to create one direct copy and one relocated copy. Extending this to additional copies is done in the same way as was done on the primary system.

Performed on the Primary System

```
root> # assumes STD are in group pr2 and already split
root> symlid -g pr2 moveall sc1
root> symmir -g sc1 -noprompt establish
root> symmir -g sc1 -i 30 verify

dba> # any steps needed to stop application
dba> db2stop

root> symmir -g sc2 -noprompt -instant split

db2> db2start
dba> # any steps needed to restart application
```

Performed on the Secondary System

```
root> importvg -ydrvg hdisk7
root> importvg -ylvg hdisk27
root> chown dba /dev/rrlv
root> mount /data
root> mount /logs

dba> db2 catalog database testdb on /data
```

This nonrelocated copy can be reestablished and used as needed. (For details see IBM Tech Report TR-74.180 or EMC White Paper JM52094, **Creating Hot Snapshots and Standby Databases with IBM DB2 Universal Database V7.2 and EMC TimeFinder.**)

Using a Relocated Database Copy on the Secondary System

Using a relocated database copy on the secondary system is done with a procedure that is essentially the same as was used on the primary system.

Performed on the Primary System

```
root> # Associate the STD volumes with the desired BCV volumes
root> symld -g sc1 moveall sc2
root> # Copy the STD volumes
root> symmir -g sc2 -noprompt establish
root> symmir -g sc2 -i 30 verify
root> # Separate the BCV copies
root> db_split dba testdb sc2
```

Performed on the Secondary System

```
root> recreatevg -ydr1vg -Yc1 -L/copy1 $dr1dk
root> recreatevg -y11vg -Yc1 -L/copy1 $11dk
root> mount /copy1/data
root> mount /copy1/logs
root> mv /copy1/data/dba /copy1/data/altdba
root> chown -R altdba /dev/rc1rlv /copy1/data /copy1/logs
```

```
altdba> cat copy1.conf
DB_NAME=testdb,testldb
INSTANCE=dba,altdba
DB_PATH=/data,/copy1/data
LOG_DIR=/logs,/copy1/logs
CONT_PATH=/dev/r1rlv,/dev/rc1rlv
altdba>
```

You can then use the copy as a snapshot:

```
altdba> # Initialize the database as a snapshot
altdba> db2start # (if not already started)
altdba> db2inidb test2db as snapshot relocate using copy21.conf
```

The relocated copy is now available for use on the secondary system.

Or you can use the copy for standby:

```
altdba> # Initialize the database in standby mode
altdba> db2start # (if not already started)
altdba> db2inidb test2db as standby relocate using copy21.conf
```

You can apply archived log files generated from the primary database (testdb) to this relocated split image database copy (test2db) on the secondary system.

The script `db_split` can be downloaded from:

```
ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest
```

Appendix A: Procedures

Symmetrix Command Access

The path to the SYMCLI software should be put into root's default profile:

```
root> export PATH=$PATH:/usr/symcli/bin
```

Additional commands are required when initializing devices:

```
root> export PATH=$PATH:/usr/lpp/Symmetrix/bin
```

AIX and Symmetrix Volume Discovery

1. Detect new Symmetrix devices on AIX:

```
root> emc_cfgmgr
```

2. Initialize communication with the Symmetrix system, and discover device paths and other device information:

```
root> symcfg discover
```

3. List the available devices so that you can determine how AIX device names are mapped to the Symmetrix volumes available to this host:

```
root> symdev list
```

Setting Up Symmetrix Device Groups for TimeFinder Operations

This defines the members of a group containing a standard volume set and a BCV volume set. You must have the same number of primary and BCV volumes, and they must be added in the order you want them associated together. The volumes to be associated must be the same size. SYMCLI commands take these group names as parameters.

Values required to run procedure:

GROUP	Name for the Symmetrix device group consisting of STD Symmetrix volumes to be mirrored with BCV volumes.
SSTART	First standard volume (three-digit hex Symmetrix device number).
SEND	Last standard volume (three-digit hex Symmetrix device number).
BSTART	First BCV volume (three-digit hex Symmetrix device number).
BEND	Last BCV volume (three-digit hex Symmetrix device number).

1. Create the group for the production volumes:

```
root> symdg -type REGULAR create GROUP
```

2. Add a range of primary devices to the group:

```
root> symld -g GROUP -RANGE SSTART:SEND addall dev
```

3. Repeat step 2 if you want to combine nonconsecutive blocks of volumes into a single STD group.

4. Associate a range of BCV devices with the group:

```
root> symbcv -g GROUP -RANGE BSTART:BEND associateall dev
```

5. Repeat step 4 if you want to combine nonconsecutive blocks of volumes into a single BCV group.

Establishing an Initial BCV Mirror

The initial establish of a BCV volume group needs to specify how the volumes are to be matched up with each other. Subsequent establish commands will simply use this previously determined matching.

Value required to run procedure:

GROUP	Name for the Symmetrix device group consisting of STD Symmetrix volumes to be mirrored with BCV volumes.
--------------	--

1. Perform an initial establish of a group of STD - BCV device pairs:

```
root> symmir -g GROUP -noprompt -full -exact establish
```

2. Repeat step 1 as needed for each group.

3. (Optional) Wait until the process completes, and then enter the following command:

```
root> symmir -g GROUP -i 30 verify
```

Splitting a Nonbusy BCV Group

Splitting a BCV volume group is quite simple when there is no database actively running on the volumes that must be kept in a consistent state for subsequent use with the BCV copy.

Value required to run procedure:

GROUP	Name for the Symmetrix device group consisting of STD Symmetrix volumes to be mirrored with BCV volumes.
--------------	--

1. Ensure the previous establish has completed.

```
root> symmir -g GROUP -i 30 verify
```

2. Split the group:

```
root> symmir -g GROUP -noprompt -instant split
```

Splitting a Busy BCV Group

Splitting a BCV volume group is done slightly differently when there is a database actively running on the volumes that must be kept in a consistent state for subsequent use with the BCV copy.

Values required to run procedure:

GROUP	Name for the Symmetrix device group consisting of STD Symmetrix volumes to be mirrored with BCV volumes.
DBA	Instance owner of the database.
DBNAME	Name of the database.

1. Ensure the previous establish has completed.

```
root> symmir -g GROUP -i 30 verify
```

2. Split the group:

```
root> db_split -g DBA DBNAME GROUP
```

The script db_split can be downloaded from:

```
ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest
```

Setting Up an Additional BCV Device Group for TimeFinder Operations

When there are multiple BCV copies of the same standard device set to manage, the procedure for creating the subsequent device groups is different from creating the first. For the subsequent BCV copies, the standard devices must be moved into the new group from the group they were in previously.

Values required to run procedure:

OGROUP	Name for the Symmetrix device group currently containing the STD Symmetrix volumes to be mirrored by the new BCV volumes.
NGROUP	Name for the new Symmetrix device group to also be mirrored with STD volumes.
BSTART	First BCV volume (three-digit hex Symmetrix device number).
BEND	Last BCV volume (three-digit hex Symmetrix device number).

1. Create the group for the new BCV volumes:

```
root> symdg -type REGULAR create NGROUP
```

2. Move the STD primary devices to the new group:

```
root> symld -g OGROUP moveall NGROUP
```

3. Associate a range of BCV devices with the group:

```
root> symbcv -g NGROUP -RANGE BSTART:BEND associateall dev
```

4. Repeat step 3 if you want to combine nonconsecutive blocks of volumes into a single BCV group.
5. Establish the BCV, wait for the establish to complete, and split it.

```
root> symmir -g NGROUP -noprompt -full -exact establish
root> symmir -g NGROUP -I 30 verify
root> symmir -g NGROUP -noprompt -instant split
```

Creating a Volume Group Containing a List of Volumes

Values required to run procedure:

VGNAME	Name for the volume group.
PPSIZE	Size of partitions to be used.
DKLIST	List of physical volumes to be included in the group.

The partition size you choose depends on the size of the hypervolumes. There is a limit of 1016 partitions per physical volume, so your partition size must be at least 0.1% of the total size you may want to use on the volume. The partition is the unit of allocation for space on the volume, so a large size will waste more space because a request is rounded up to the partition size. For example, with hypervolumes of 4 GB, the default partition size of 4 MB would be too small. Using 8 MB would work or, if the entire partition is going to be used so there is no concern for wasting a partial partition, a larger size could be convenient.

```
root> mkvg -f -yVGNAME -sPPSIZE DKLIST
```

Creating a Logical Volume for Journalled File System

Values required to run procedure:

LVNAME	Name for the logical volume.
MAXPP	Maximum number of partitions to be used.
VGNAME	Name of the volume group to contain the logical volume.
DKLIST	List of physical volumes to be included in the logical volume.

Specify the number of partitions to be available to the logical volume (which is set here to almost all of the available space). We set the space initially to 1, so that when a journalled file system is built, the logical volume to be used for the journal is placed in the middle of the available space for improved speed. This is not especially critical for volumes that are on a Symmetrix system, but it does not hurt and can have some beneficial effect.

```
root> mklv -yLVNAME -tjfs -xMAXPP VGNAME 1 DKLIST
```

Creating a Journalled File System

Values required to run procedure:

LVNAME	Name of the logical volume.
MNTPT	Mount point where the file system will be used.
AUTO	Specifies whether the file system is to be mounted on reboot(either yes or no); usually yes for the primary system and no for the secondary system.
ISIZE	Size for inode blocks.
FSSIZE	Final size for the file system (in 512 byte blocks).
DBA	User ID for the account that will own the database.

A file system needs a mount point where it will appear in the computer's directory structure. The size of an inode block controls the number of files (and directories and devices and so on) that can be created on that file system. An alternate approach is to use `smit fs` to create file systems.

```
root> crfs -vjfs -dLVNAME -mMNTPT -AAUTO -prw -tno -a nbpi=ISIZE
root> chfs -a size=FSSIZE MNTPT
root> mount MNTPT
root> chown DBA MNTPT
```

Creating a Logical Volume to Contain Raw Disks

Values required to run procedure:

LVNAME	Name for the logical volume.
MAXPP	Maximum number of partitions to be used.
VGNAME	Name of the volume group to contain the logical volume.
ACTPP	Initial number of partitions to be used.
DKLIST	List of physical volumes to be included in the logical volume.
DBA	User ID for the account that will own the database.

Specify the number of partitions available to the logical volume. The space is allocated immediately here. No other logical volumes are involved for a raw volume, so no special arrangements are required to control the positioning within the volume group.

```
root> mklv -yLVNAME -xMAXPP VGNAME ACTPP DKLIST
root> chown DBA /dev/rLVNAME
```

Creating a DB2 Database in a Journalled File System

Values required to run procedure:

DBNAME	Name of the database to be created.
PATH	Location of the database.

1. Start IBM DB2 UDB (if it is not already running):

```
dba> db2start
```

2. Create the database on the file system:

```
dba> db2 create db DBNAME on PATH
```

Configuring a Raw Device as DMS in a Database

A raw device can be used to hold data within a database as DMS (Database- Managed Storage), either for the entire content of the database or to augment the data stored in the previously created file system space, which DB2 calls SMS (System- Managed Storage).

Values required to run procedure:

DBNAME	Name of the database.
TSNAME	Name to give the tablespace that will contain the raw device.
TBNAME	Name of the table in the tablespace.
TBDEF	DB2 table definition.
DVPATH	Pathname of the raw device container.
DVSIZE	Amount of space available on the device.

1. Establish a connection to the database:

```
dba> db2 connect to DBNAME
```

2. Use " " on the next command so that the quotes and parentheses will not need to be escaped to protect them from interpretation by the shell:

```
dba>db2 "create tablespace TSNAME managed by database using \  
      ( device 'DEVPATH' DEVSIZE )"
```

3. Create table(s) in the tablespace:

```
dba> db2 "create table TBNAME ( TBDEF ) on TSNAME"
```

Setting the Location of Logs

The default location for the DB2 log files is in the `SQLLOGDIR` directory, which is a relative path under the database path used during the create database command. You would omit setting the location of logs if you were not using a separate BCV group for the logs and the log path was contained under the database path.

Values required to run procedure:

DBNAME	Name of the database.
LPATH	Location for the logs.

Modify the default location of the DB2 log files:

```
dba> db2 update db cfg for DBNAME using NEWLOGPATH LPATH
```

Importing a Nonrelocated Disk Configuration to a Secondary System

The volume manager setup, file system setup, and database setup have already been done on the STD volumes on the primary system, and copied to the BCV volumes; so the secondary system merely has to learn about them. Since there is no relocation, they can be used as they are. Using the AIX volume manager, this step is simple.

Values required to run procedure:

VGNAME	Name of the volume group being imported.
DISK	Device name of the first disk in the group.
DBUSER	Instance owner of the database.
RAWDEV	Name of a raw device (if any) in the volume.

1. Import volume group definition from the physical volumes:

```
root> importvg -yVGNAME DISK
```

2. If a raw device exists, it needs to have permissions set correctly (the contents of the file systems will have the permissions already set from the mirroring of STD devices on the primary system).

```
root> chown DBOWNER RAWDEV
```

Cataloging a Nonrelocated Database on the Secondary System

The database is already present on the imported disk volumes, but IBM DB2 UDB on the secondary system does not know about the imported volumes yet. The internal information in the database is consistent with the (nonrelocated) position of the imported data, so no extra change is required to deal with that.

Values required to run procedure:

DBNAME	Name of the database.
PATH	Location of the database.

Make the database known to the DB2 instance:

```
db2> db2 catalog database DBNAME on PATH
```

Duplicating the Disk Configuration to use an Additional BCV Copy

The volume manager setup, file system setup, and database setup have already been done on the STD volumes and copied to the BCV volumes. We need to create an identical layout, but cannot just use the existing layout because the names would conflict with the original system.

Values required to run procedure:

VGNAME	Name of the volume group being created.
DISKS	Device names of the disks in the group.
DIRPRE	The directory prefix to be used to contain the mount points of the copy.
DEVPRE	The prefix to be used to modify the device names from their original values.
ALTDBA	The instance name of the new database copy (which must be different from the instance name of the original database).
DATA	The mount point of the original database data.
LOGS	The mount point of the original database logs.
RAWDEV	The device name of the original raw device (omitting the leading letter "r").

1. Create a new volume group definition from the physical volumes that is consistent with the original layout:

```
root> recreatevg -yVGNAME -yDEVPR -lDIRPRE DISKS
root> mount DIRPRE/DATA
root> mount DIRPRE/LOGS
```

Note: The mount points of the newly created volume group are placed under the provided prefix but are otherwise the same.

2. The files (and a raw device if it was used), must be given to the new instance.

```
root> chown -R ALTDBA DIRPRE/DATA DIRPRE/LOGS rDEVPRERAWDEV
```

Database Relocation File for an Additional BCV Copy of a Database

The database is already present on the imported disk volumes, but IBM DB2 UDB does not know about it yet. Furthermore, the internal information within the copied data still refers to the positioning of the original database and not the relocated positioning of the copy. A configuration file is required for the procedure that tells IBM UDB DB2 about the new copy.

Values required to run procedure:

ODBNAM	Original name of the database.
NDBNAM	Name of the new copy.
OINST	Original instance name.
NINST	New instance owner name.
OPATH	Location of the original database.
NPATH	Location of the new copy.
OLOGS	Location of the original logs.
NLOGS	Location of the new logs.
OCONT	Location of an original container.
NCONT	Location of the copied container.

Create a configuration file with the relocation information:

```
DB_NAME=ODBNAM ,NDBNAM
INSTANCE=OINST ,NINST
DB_PATH=OPATH ,NPATH
LOG_DIR=OLOGS ,NLOGS
CONT_PATH=OCONT ,NCONT
```

The CONT_PATH entry can be repeated if your database has multiple containers, or it can be omitted if your database has no containers or if they are all below the database's root directory, DB_PATH.

The LOG_DIR entry can be omitted if the logs are kept under the DB_PATH directory.

There can also be an entry for NODENUM that can be used in an EEE environment.

Relocating an Additional BCV Copy of a Database

The database must be updated using the configuration file to be consistent with its relocated position.

Values required to run procedure:

OINST	Original instance name.
NINST	New instance owner name.
NPATH	Location of the new database copy.
NDBNAM	Name of the new database.
CONF	Relocation configuration file.

1. Put the database into the right path for the instance:

```
root> mv NPATH/OINST NPATH/NINST
```

2. Tell DB2 about the new database and have it relocate the internal information:

```
altdba> db2start  
altdba> db2inidb NDBNAM as snapshot relocate using CONF
```

A copied database can be used for many purposes. Such purposes often must be repeated on a fresh copy of the database. For example, a monthly reporting process that uses a copy of the database so that no changes will be applied while the report is being collected is repeated each month on a fresh copy of the database.

Preparing for Reestablish on the Secondary System

If the BCV volumes you want to establish are being used on the secondary system, they must be freed up. You have to stop any processes that are using the volumes, including AIX's use of file systems mounted from any of the volumes. This whole step is specific to the way you are using the volumes on your system; we do not provide a general procedure. You do not need to take special action to stop these processes *cleanly*. Because the volumes will be established as BCV copies of the primary system, the data on them from the current processes on the secondary system is totally overwritten. So, you may want to kill the processes, an action that would be unthinkable on the primary system with a live database.

Value required to run procedure:

FILESYS	Name of a file system to unmount.
DBNAM	Name of the database (only needed if there are other databases in the same instance that must be retained – see version 2 below)

1. Find any DB2 applications running in the instance:

```
dba> db2 list applications
```

2. Stop running DB2 applications. This may require application-specific procedures. If so, perform them before the following:

(Version 1 – no other database are running in this instance)

```
dba> db2stop force
```

(Version 2 – there are other databases running in this instance)

Stop the applications for this database (use the application handles, *H1*, *H2*, ... displayed by the `list applications` command) and uncatlog the database (*DBNAM*):

```
dba> db2 'force application ( H1, H2, ... )'
dba> db2 uncatlog database DBNAM
dba> db2 terminate
```

3. Unmount the file systems:

```
root> umount FILESYS
```

4. Repeat step 3 for each file system.

Reestablishing on the Primary System

Value required to run procedure:

GROUP	Name of a device group that will be established.
--------------	--

Establish each group of devices:

```
root> symmir -g GROUP -noprompt establish
```

Removing a Database Copy: Drop from AIX

Values required to run procedure:

DIRPRE	Prefix directory for copied data and logs.
DBPATH	Path to the original database.
LOGS	Path to the original logs.
DVGNAM	New data volume group name.
LVGNAM	New logs volume group name.

Make AIX forget about the database copy that is about to be over-written:

```
root> umount /DIRPRE/DBPATH
root> umount /DIRPRE/LOGS
root> varyoffvg DVGNAM
root> varyoffvg LVGNAM
root> exportvg DVGNAM
root> exportvg LVGNAM
```

Updating the Database BCV Copy from the Standard

Values required to run procedure:

CURDG	Symmetrix device group with which the standard is currently associated.
BCVDG	Symmetrix device group containing the BCV volumes.
DBA	Instance owner of the database.
TESTDB	Database name.

```
root> # Run next command only if CURDG is different from BCVDG.
root> # Skip next command if they are the same.
root> symld -g CURDG moveall BCVDG

root> symmir -g BCVDG -noprompt establish
root> symmir -g BCVDG -i 30 verify

root> db_split -g TESTDB DBA BCVDG
```

The script `db_split` can be downloaded from:

ftp://ftp.emc.com/pub/symm3000/DB2/db_split_latest