Hitachi Data Systems
IBM

Using Hitachi® Data Systems 7700E
Open ShadowImage™ (HOMRCF)
And Open Remote Copy (HORC)
with
IBM® DB2® Universal Database™ Enterprise-
Extended Edition Version 7.2
(TR-74-182)

March 29, 2002

Aslam Nomani – IBM Toronto Laboratory
Enzo Cialini – IBM Toronto Laboratory
Using Hitachi Data Systems HOMRCF and HORC with DB2 UDB Enterprise Edition
**Trademarks**

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX
DB2
DB2 Universal Database
IBM

The following terms are trademarks or registered trademarks of Hitachi Corporation in the United States, or other countries, or both: Hitachi, ShadowImage.

UNIX and UNIX-based trademarks and logos are trademarks or registered trademarks of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

The furnishing of this document does not imply giving license to any IBM or Hitachi 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.
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>V</td>
</tr>
<tr>
<td>ABOUT THE AUTHORS</td>
<td>VI</td>
</tr>
<tr>
<td>ITIRC KEYWORDS</td>
<td>VII</td>
</tr>
<tr>
<td>DB2 UDB SUSPEND I/O OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>DB2 UDB COMMAND DESCRIPTION</td>
<td>2</td>
</tr>
<tr>
<td>SUSPEND I/O COMMAND</td>
<td>2</td>
</tr>
<tr>
<td>RESUME I/O COMMAND</td>
<td>2</td>
</tr>
<tr>
<td>DB2INIDB COMMAND</td>
<td>2</td>
</tr>
<tr>
<td>RESTART WITH WRITE RESUME COMMAND</td>
<td>2</td>
</tr>
<tr>
<td>OVERVIEW OF HITACHI OPEN SHADOWIMAGE (HOMRCF)</td>
<td>3</td>
</tr>
<tr>
<td>OVERVIEW OF HITACHI OPEN REMOTE COPY (HORC)</td>
<td>4</td>
</tr>
<tr>
<td>OVERVIEW OF HITACHI COMMAND CONTROL INTERFACE (CCI)</td>
<td>5</td>
</tr>
<tr>
<td>CCI COMMAND DESCRIPTION</td>
<td>7</td>
</tr>
<tr>
<td>PAIR CREATE COMMAND</td>
<td>7</td>
</tr>
<tr>
<td>PAIR SPLIT COMMAND</td>
<td>7</td>
</tr>
<tr>
<td>PAIR RESYNCHRONIZATION COMMAND</td>
<td>7</td>
</tr>
<tr>
<td>PAIR EVENT WAITING COMMAND</td>
<td>8</td>
</tr>
<tr>
<td>PAIR MONITORING COMMAND</td>
<td>8</td>
</tr>
<tr>
<td>PAIR DISPLAY COMMAND</td>
<td>8</td>
</tr>
<tr>
<td>HORCM START COMMAND</td>
<td>8</td>
</tr>
<tr>
<td>HORCM SHUTDOWN COMMAND</td>
<td>8</td>
</tr>
<tr>
<td>DB2 UDB AND HITACHI DATA SYSTEMS SAMPLE CONFIGURATION</td>
<td>9</td>
</tr>
<tr>
<td>CONFIGURING A TWO HOST SYSTEM (ONE P-VOL, ONE S-VOL)</td>
<td>12</td>
</tr>
<tr>
<td>SNAPSHOT DATABASE SCENARIO</td>
<td>22</td>
</tr>
<tr>
<td>STANDBY DATABASE SCENARIO</td>
<td>24</td>
</tr>
<tr>
<td>SPLIT-MIRROR ONLINE BACKUP SCENARIO</td>
<td>28</td>
</tr>
<tr>
<td>SPLIT-MIRROR RESTORE SCENARIO</td>
<td>32</td>
</tr>
<tr>
<td>INCREMENTAL RESTORE (PAIRRESYNCHRONIZATION VS. PAIRCREATE)</td>
<td>35</td>
</tr>
<tr>
<td>APPENDIX A – SPLITTING PAIRS WHEN THE PRIMARY SERVER IS DOWN</td>
<td>36</td>
</tr>
<tr>
<td>SPLITTING SHADOWIMAGE PAIRS</td>
<td>36</td>
</tr>
<tr>
<td>SPLITTING REMOTE COPY PAIRS</td>
<td>38</td>
</tr>
</tbody>
</table>
Using Hitachi Data Systems HOMRCF and HORC with DB2 UDB Enterprise Edition
Abstract

IBM® DB2® Universal Database™ is the industry’s first multimedia, Web-ready relational database management system that is strong enough to meet the demands of large corporations and flexible enough to serve medium-sized and small e-businesses. DB2 Universal Database (UDB) combines integrated power for business intelligence, content management, and e-business with industry-leading performance and reliability. When coupled with the Hitachi® Open ShadowImage™ (HOMRCF) and Hitachi Open Remote Copy (HORC) features of the Hitachi Data Systems Storage Subsystem strengthens the solution.

About the Authors

Aslam Nomani has been working with the Database Technology (DBT) team in the IBM Toronto Laboratory for five years. For the past four years, he has worked in the DB2 Universal Database (UDB) System Verification Test department. Aslam has worked extensively in testing DB2 Universal Database in high availability environments. He is currently a team lead within the DBT test organization.

Enzo Cialini has been working with Database Technology at the IBM Toronto Laboratory for ten years and is certified in DB2 UDB Database Administration and DB2 UDB Database Application Development. He is also an Advanced Technical Expert in DB2 for DRDA, and an Advanced Technical Expert in DB2 for Clusters. He is currently responsible for managing the DB2 UDB System Verification Test department, with a focus on high availability, and has been involved with DB2 in high availability environments for many years. His experience ranges from implementing and supporting numerous installations to consulting.
ITIRC Keywords

DB2
DB2 UDB
Availability
Suspend I/O
Mirror
Hitachi
DB2 UDB Suspend I/O Overview

DB2 Universal Database Version 7.2 has implemented suspend I/O technology, which provides the capability to use split-mirroring technology while DB2 is online. Suspend I/O supports continuous system availability by providing a full implementation for splitting a mirror (pair) without shutting down the database. The split copy of the database can be utilized to do such tasks as the following:

1) Provide a transactionally consistent snapshot of the database at the current point in time. This database can be used to offload user queries that do not need the most current version of the database.

2) Provide a standby database that can be accessed as a disaster recovery strategy if the primary database is not available. All logs from the primary database will be applied to the secondary database so it will represent a transactionally consistent version of the primary database at the time of the failure.

3) Provide the ability to offload backups from the primary database. A DB2 backup can be performed on the secondary database system. The DB2 backup can then be restored on either the primary system or on another system. Rollforward can then be issued to bring the database to a particular point in time, or till the end of the logs is reached.

4) Provide the ability for a quick database restore. The mirror (pair) can be resynchronized so the primary copy is restored to the initial data at the time of the split. Rollforward can then be issued on the primary database to bring the database to a particular point in time, or till the end of the logs is reached.
**DB2 UDB Command Description**

**Suspend I/O Command**
The suspend command (`db2 set write suspend for database`) suspends all write operations to a DB2 UDB database partition (that is, tablespaces and log files). Read operations are not suspended and are thus allowed to continue. Applications can continue to process insert, update, and delete operations utilizing the DB2 buffer pools. A database connection is required for issuing the suspend command. It is recommended to maintain the current session for executing the subsequent resume command.

**Resume I/O Command**
The resume command (`db2 set write resume for database`) resumes all write operations to the suspended DB2 UDB database partition. A database connection is required for issuing the resume command.

**db2inidb Command**
The `db2inidb` command (`db2inidb <db_alias> as < snapshot | standby | mirror >`) is required to initialize the copy of the suspended database partition. You do not require a database connection in order to execute this command. It can be used in the following three cases:

1) Snapshot can be applied to the secondary copy, which will put the secondary database into a transactionally consistent state.

2) Standby can be applied to the secondary copy to put the secondary database into a rollforward-pending state. DB2 logs from the primary database can then be applied to the secondary database, if required.

3) 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, if required.

**Restart with Write Resume Command**
The restart with write resume command (`db2 restart db <db_alias> write resume`) can be utilized to resume write operations on a suspended database partition that experienced an abnormal termination.

For further information regarding DB2 UDB’s suspend I/O functionality, refer to the *DB2 UDB documentation* and *Release Notes* provided with Version 7.2 or Version 7.1 FixPak 3.
Overview of Hitachi Open ShadowImage (HOMRCF)

The Hitachi Open ShadowImage feature of the 7700E RAID storage subsystem enables you to maintain multiple subsystem-internal copies of open-systems logical units (LUs) for purposes such as data backup or data duplication. The RAID-protected Open ShadowImage duplicate volumes are created within the same 7700E subsystem as the primary volume at hardware speeds. Once established, Open ShadowImage operations continue unattended to provide periodic, point-in-time data backups. Open ShadowImage operations are nondisruptive and allow the primary volume of each Open ShadowImage pair to remain online to all hosts for both read and write I/O operations. Hitachi Data Systems further enhances usability through a resynchronization capability that reduces data duplication requirements and increases user productivity.

Open ShadowImage (HOMRCF) is a storage-based hardware solution for duplicating logical volumes that reduces backup time and provides point-in-time backup. The HOMRCF primary volumes (P-VOLs) contain the original data, and the HOMRCF secondary volumes (S-VOLs) contain the duplicate data. The user can choose to make one, two, or three copies of each P-VOL (up to three S-VOLs). And since each S-VOL is paired with its P-VOL independently, each S-VOL can be maintained as an independent copy set that can be split, suspended, resynchronized, and deleted separately from the other S-VOLs assigned to the same P-VOL.
Overview of Hitachi Open Remote Copy (HORC)

The HORC feature enables you to create and maintain multiple duplicate copies of all user data across multiple Hitachi 7700E storage subsystems. This allows the primary disk subsystem and secondary disk subsystem to be located at different physical sites to enhance a disaster recovery strategy. During normal HORC operations, the primary volumes remain online to all hosts and continue to process both read and write I/O operations. In the event of a disaster or system failure, the secondary copy of the data can be rapidly invoked to allow recovery with a very high level of data integrity. HORC can be performed in both synchronous mode (HORC) and asynchronous mode (HOARC). The examples in this paper will utilize the asynchronous mode of Remote Copy.
Overview of Hitachi Command Control Interface (CCI)

CCI Instances
Each copy of CCI, on a UNIX or Windows NT platform, is known as a CCI instance. Each instance uses a defined configuration file to manage volume relationships while maintaining awareness of the other CCI instances.

There are four typical CCI topologies:
- One host connected to one 7700E
- One host connected to two 7700Es
- Two hosts connected to one 7700E
- Two hosts connected to two 7700Es

Configuration File

The CCI configuration file (horcm.conf) is a text file that provides a definition of connected hosts, and the volumes and groups known to the CCI instance. The configuration file contains the following four areas that define CCI settings: HORCM_MON, HORCM_CMD, HORCM_DEV, and HORCM_INST.

HORCM_MON
The MON (monitor) section of the configuration file defines CCI monitoring and communicating parameters. The host IP address, service name, poll rate and timeout value are all set in the MON section of the configuration file.

HORCM_CMD
The CMD (command) section of the configuration file defines the dedicated command device on the 7700E subsystem. You can define more than one command device in this section to provide failover functionality, should the original command device become inoperable.

Note: The command device must also be defined from the 7700E subsystem, either through the Remote Console’s LUN Manager software (refer to the Hitachi Freedom 7700E Remote Console User’s Guide (BO-98DD847)) or by your Hitachi Data Systems representative.

HORCM_DEV
The DEV (device) section of the configuration file defines the addresses in the 7700E of the physical volumes corresponding to the paired logical volume names. The DEV section also defines the primary or secondary volumes used for remote copy. Volume pairs can be assigned names for easy pair reference, and volume names can be assigned group names. Each group name is a unique name discriminated by a host that uses the volumes. The CCI software can use either the device names or the defined group names for the associated device name. This flexibility allows you to manage either individual pairs or groups of pairs.
HORCM_INST
The INST (instance) section of the configuration file defines the network settings that correspond to the defined group names in the DEV section of the configuration file. The group name, IP address of the second host, and the function name are set in the INST section of the configuration file. The instance information is used in the event of a failure at either the primary or secondary host. The instance definition makes the CCI software aware of another active instance (either on the same host or on a different host, depending on your configuration).
CCI Command Description

Pair Create Command
The pair create command (paircreate) generates a new volume pair from two unpaired volumes. The pair create command creates either a paired logical volume or group. Using the pair create command, you can specify which server has the primary volume. If the vl option is specified, the server issuing the command has the primary volume. If the vr option is specified, the remote server has the primary volume.

Note: The pair create command terminates before copying of the entire volume data is completed (except in the No Copy mode). Use the pair display command to verify that the data copy has completed. The execution log file also allows you to verify the success of the pair create command.

Pair Split Command
The pair split command (pairsplit) stops updating the secondary volume while maintaining the pairing status. When this command is issued, a read or read/write access to the secondary volume is enabled (depending on the selected options). The primary volume’s server is automatically detected by the pair split command, so the server does not need to be specified in the pair split command parameters. Paired volumes are split as soon as the pair split command is issued. This command must be issued after write commands to the paired volume have completed.

Note: The pair split command terminates after verifying that the volume status has been changed (according to the pair split command). When pair splitting is specified, acceptance of a write request to the primary volume depends on the fence level.

Pair Resynchronization Command
The pair resynchronization command (pairresync) updates the secondary volume to be synchronized with the primary volume and re-establishes the pair. If no data has been written to the secondary volume, differential data is copied. If data has been written to the secondary volume, the entire primary volume data is copied. Pair resynchronization can be specified while the primary volume is being accessed. When the pair resynchronization command is issued, any host write access to the secondary volume is disabled. The primary volume’s server is automatically detected by the resynchronization command, so the server does not need to be specified in the pair resync command parameters.

Note: The pair resync command terminates before resynchronization of the secondary volume is completed. Use the event waiting command to check completion of resynchronization. The transition of the paired volume status is judged by the status of the primary volume. The fence level specified in the pair generation command is left unchanged.
**Pair Event Waiting Command**
The pair event waiting command (**pairevtwait**) is used to check the status of pair creation and pair resynchronization while waiting for their completion. It waits (sleeps from the viewpoint of the processing) until the paired volume status becomes identical to a specified status and completes. The primary volume’s server is automatically detected by the pair event waiting command, so the server does not need to be specified in the pair event waiting command parameters.

**Note:** The pair event waiting command waits until the specified status is established. It terminates abnormally if an abnormal paired volume status is detected. The transition of the paired volume status is judged by the status of the primary volume. If the event waiting command is issued for a group, the command waits until the status of each volume in the group becomes identical to the specified status. When the event waiting command with the "No Wait" option is issued for a group, the status is returned if the status of each volume in the group is identical.

**Pair Monitoring Command**
The pair monitoring command (**pairmon**) is part of the CCI daemon and it reports the status of each volume pair. If an error occurs or if the pair status changes, the pair monitoring command issues an error message. The CCI software supports the error monitoring and configuration confirmation commands for linkage with the system operation management of the UNIX server.

**Pair Display Command**
The pair display command (**pairdisplay**) displays the pairing status, allowing you to verify completion of the pair generation or pair resynchronization operations. The pair display command is also used to confirm the configuration of the paired volume connection path (the physical link of paired volumes and servers).

**HORCM Start Command**
The HORCM start command (**horcmstart.sh**) is a shell script that starts CCI activities. This shell script also sets the environment variables for the CCI software in the following files: HORCM_CONF, HORCM_LOG, and HORCM_LOGS.

**HORCM Shutdown Command**
The HORCM shutdown (**horcmshutdown.sh**) command is a shell script for stopping the CCI software. If no environment variables are set in the execution environment, the set environment command (**setenv**) sets specified environment variables individually. The un-set environment command (**usethen**) deletes specified environmental variables, the environment command (**env**) displays the environmental variables, and the sleep command suspends the server for the specified time.
DB2 UDB and Hitachi Data Systems Sample Configuration

The following diagram illustrates ShadowImage and Remote Copy functionality with DB2 UDB. These examples will also identify when the ShadowImage and Remote Copy commands differ. ShadowImage commands will be referenced using HOMRCF, and asynchronous remote copy commands will be referenced using HOARC. The DB2 UDB EEE instance can span one or more physical machines.

![Diagram](image.png)

Figure 1: Conceptual Overview: Two AIX® hosts, one as the primary server with two DB2 database partitions, the other as the backup server with two DB2 database partitions. One CCI instance will be configured on each system.

DB2 Universal Database Version 7.2 for AIX was used to document these procedures. DB2 UDB suspend I/O capabilities are available on all platforms that DB2 UDB supports.

Note: The Hitachi 7700E was used to document the following examples. The same DB2 UDB commands are supported on the Hitachi 9900 using corresponding CCI commands.

ShadowImage and Remote Copy Installation

The Hitachi Data Systems (HDS) representative installs ShadowImage (HOMRCF) and the Remote Copy software on the HDS 7700E.
From each AIX server, run `cfgmgr` to define SCSI paths to HDS 7700E.

**Command Control Interface Installation on AIX**

Install RaidManager Command Control Interface on each AIX server and set the HORCC_MRCF environment variable to 1 if ShadowImage functionality will be utilized. If Remote Copy is being utilized, ensure the HORCC_MRCF environment variable is not set:

```
export HORCC_MRCF=1  (for HOMRCF)
```

The AIX installation of the Command Control Interface software is done by the customer, with help as needed from the Hitachi Data Systems representative.

1. Login with root access to AIX.
2. Create a root level directory called /HORCM:

   ```
   mkdir /HORCM
   ```

3. Change to the HORCM directory and create a temp directory, then change to the temp directory by entering:

   ```
   cd /HORCM
   mkdir tmp
   cd tmp
   ```

4. Extract the 3 cpio archive files:
   For floppy disks enter:
   ```
   cpio -ID /dev/floppy0
   ```
   **Note:** Repeat this command for each disk.

   For CD enter:
   ```
   cat /cdrom/aix/fd1 | cpio –id
   cat /cdrom/aix/fd2 | cpio –id
   cat /cdrom/aix/fd3 | cpio –id
   ```

5. Change to the HORCM directory:

   ```
   cd /HORCM
   ```

6. Move the /HORCM/tmp/HORCM directory to the root level /HORCM directory:

   ```
   mv /HORCM/tmp/HORCM /HORCM
   ```
7. Execute the horcminstall.sh script:

   ./horcminstall.sh

8. Edit the /etc/services file and add the following information to the file. This port will be used by a CCI instance:

   horcm 11000/udp     #HORCM port

   **Note:** The port number (xxxx/udp) used here is an example. To find a suitable port number, list the port numbers that are currently in use by viewing the /etc/services file. Choose any number that is not in use (the port number must be between 5000 and 65535).

9. Make a backup copy of the horcm.conf file:

   cd /etc
   cp horcm.conf horcmbackup.conf

10. Determine SCSI paths and disks to be used for the DB2 database and logs.
11. Document corresponding ldev’s that will be used for AIX logical volumes.
Configuring a Two Host System (One P-Vol, One S-Vol)

Overview
- Prepare ShadowImage or Remote Copy on both servers.
- Install DB2 for AIX on both servers.

This diagram illustrates the configuration used in the examples that follow using ShadowImage:

Figure 2: ShadowImage Configuration

The key points to note from Figure 2 are:

- The host name of the Primary server is MACHA.
- MACHA contains DB2 partitions 0 and 1.
- The host name of the Backup server is MACHB.
- MACHB contains DB2 partitions 0 and 1.
- The pairs created are db2dataN0, db2dataN1, db2logN0 and db2logN1.
- The Primary server utilizes the P-Vols while the Backup server utilizes the S-Vols.
- The Primary server accesses the devices LU0, LU1, LU2 and LU3 on port CL1-E. These devices are visible to AIX as hdisk3, hdisk4, hdisk5 and hdisk6, respectively.
- The Backup server accesses the devices LU0, LU1, LU2 and LU3 on port CL2-E. These devices are visible to AIX as hdisk4, hdisk5, hdisk6 and hdisk7, respectively.
- Each half of the pair is a physically distinct disk.
- The P-Vols and S-Vols reside in the same storage subsystem.

This diagram illustrates the configuration used in the examples that follow using Remote Copy.

![Diagram](image)

Figure 3: Remote Copy Configuration

The key points to note from Figure 3 are:

- The host name of the Primary server is MACHA.
- The host name of the Backup server is MACHB.
- The pairs created are db2dataN0, db2dataN1, db2logN0 and db2logN1.
- The Primary server utilizes the P-Vols while the Backup server utilizes the S-Vols.
- The Primary server accesses the devices LU0, LU1, LU2 and LU3 on port CL1-E. These devices are visible to AIX as hdisk3, hdisk4, hdisk5 and hdisk6, respectively.
- The Backup server accesses the devices LU0, LU1, LU2 and LU3 on port CL2-E. These devices are visible to AIX as hdisk4, hdisk5, hdisk6 and hdisk7, respectively.
- Each half of the pair is a physically distinct disk.
- The P-Vols and S-Vols reside in different storage subsystems.

The above examples illustrate a Primary server that contains two partitions. DB2 also supports configurations where the partitions are spread across multiple machines and connected to one or more 7700e disk subsystems.

**Note:** All Hitachi commands and AIX commands are performed from the root userid. The DB2 install and DB2 instance creation are also performed by root. The rest of the DB2 commands are executed from the db2inst1 ID, which is the DB2 instance owning ID.

1. Verify that the /HORCM directory is on both the Primary and Backup servers.
2. Verify that the HORCC_MRCF environment variable is set to 1 only if ShadowImage is being used.
3. Verify /etc/services on each machine has a horcm entry.
4. Configure the horcm.conf file on the Primary and Backup servers.

Primary server /etc/horcm.conf example:

```
HORCM_MON
#ip_address   service   poll(10ms)   timeout(10ms)
macha         horcm     1000          3000

HORCM_CMD
#dev_name     dev_name     dev_name
/dev/rhdisk11

HORCM_DEV
#dev_group   dev_name   port#   TargetID  LU#   MU#
db2dataN0    dataN0     CL1-E    0       0
/db2dataN1    dataN1     CL1-E    0       1
/db2logN0     logN0      CL1-E    0       2
/db2logN1     logN1      CL1-E    0       3

HORCM_INST
#dev_group   ip_address   service
```
db2dataN0   machb   horcm
db2dataN1   machb   horcm
db2logN0    machb   horcm
db2logN1    machb   horcm

Backup server /etc/horcm.conf example:

HORCM_MON
#ip_address     service     poll(10ms)     timeout(10ms)
machb           horcm       1000            3000

HORCM_CMD
#dev_name           dev_name           dev_name
/dev/rhdisk28

HORCM_DEV
#dev_group         dev_name        port#    TargetID    LU#    MU#
db2dataN0         dataN0          CL2-E     0           0
/db2dataN1         dataN1          CL2-E     0           1
/db2logN0          logN0           CL2-E     0           2
/db2logN1          logN1           CL2-E     0           3

HORCM_INST
#dev_group        ip_address       service
db2dataN0         macha           horcm
db2dataN1         macha           horcm
db2logN0          macha           horcm
db2logN1          macha           horcm

Note: A multiple LUN configuration for the DB2 data and logs would have a
HORCM_DEV section defining multiple dev_name’s for each dev_group, as shown
in the following example:

HORCM_DEV
#dev_group         dev_name        port#    TargetID    LU#    MU#
db2dataN0         data1N0         CL2-E     0           0
db2dataN0         data2N0         CL2-E     0           1
db2dataN0         data3N0         CL2-E     0           2

5. Startup RaidManager from each machine:
   horcmstart.sh

Errors will be logged in /HORCM/log/curlog directory.

6. Create synchronized pairs from the Primary Server (Primary --> Backup):
   HOMRCF: paircreate –g db2dataN0 –vl
paircreate –g db2logN0 –vl
paircreate –g db2dataN1 –vl
paircreate –g db2logN1 -vl

HOARC:
paircreate –f async –g db2dataN0 –vl
paircreate –f async –g db2logN0 –vl
paircreate –f async –g db2dataN1 –vl
paircreate –f async –g db2logN1 -vl

Note: If these commands were executed from the backup server, then the direction flag would need to be changed from –vl to –vr. Otherwise, the direction of the copy would be reversed.

7. Check status of pairs:
pairdisplay –g db2dataN0
pairdisplay –g db2logN0
pairdisplay –g db2dataN1
pairdisplay –g db2logN1

When the status is ‘PAIR” we have a current mirrored DB2 system seen only from the Primary server. Pairs are now in a synchronized state.

8. On the Primary server, create volume groups, logical volumes, and file systems to be used by DB2. Then change the size of the file system, mount the file system, and change ownership of the file system to the DB2 instance owner. Below are the file systems that will be created:
   /db2data/db2inst1/NODE0000 – data file system for Partition 0
   /db2log/db2inst1/NODE0000 – log file system for Partition 0
   /db2data/db2inst1/NODE0001 – data file system for Partition 1
   /db2log/db2inst1/NODE0001 – log file system for Partition 1

Note: The home directory for the DB2 instance owner will be created independently on each machine and will not utilize any pairs. DB2 archive logs do not have to be placed in a pair. They are treated and stored normally, as part of your recovery strategy and plan.

Example Volume Group, Logical Volume, and File System creation or changes on the Primary server:

Create Volume Groups:
# smit vg
->Add a Volume Group
Alternatively, issue the following command:

```
mkvg -f -y vgdb2dataN0 -s 64 hdisk3
```

Create the remaining three volume groups from either smit or the command line:

```
mkvg -f -y vgdb2dataN1 -s 64 hdisk4
mkvg -f -y vgdb2logN0 -s 64 hdisk5
mkvg -f -y vgdb2logN1 -s 64 hdisk6
```

Create Logical Volumes:

```
#smiv lv
->Add a Logical Volume
 ->VOLUME GROUP name [vgdb2dataN0]
```

Alternatively, issue the following command:

```
mklv -y lvdb2dataN0 vgdb2dataN0 1
```

Create the remaining three logical volumes from either smit or the command line:

```
mklv -y lvdb2dataN1 vgdb2dataN1 1
mklv -y lvdb2logN0 vgdb2logN0 1
mklv -y lvdb2logN1 vgdb2logN1 1
```
Create File Systems:
#smit fs
- Add / Change / Show / Delete File Systems
- Journaled File Systems
  - Add a Journaled File System on a Previously Defined Logical Volume
  - Add a Standard Journaled File System

Alternatively, issue the following command:
```
crfs -v jfs -d lvdb2dataN0 -m /db2data/db2inst1/NODE0000 -A yes -p rw
```

Create the remaining three logical volumes from either smit or the command line:
```
crfs -v jfs -d lvdb2dataN1 -m /db2data/db2inst1/NODE0001 -A yes -p rw
```
```
crfs -v jfs -d lvdb2logN0 -m /db2log/db2inst1/NODE0000 -A yes -p rw
```
```
crfs -v jfs -d lvdb2logN1 -m /db2log/db2inst1/NODE0001 -A yes -p rw
```

Change the size of the file system (it is changed to ~7 GB in this example):
```
chfs -a size=14000000 /db2data/db2inst1/NODE0000
chfs -a size=14000000 /db2data/db2inst1/NODE0001
chfs -a size=14000000 /db2log/db2inst1/NODE0000
chfs -a size=14000000 /db2log/db2inst1/NODE0001
```

Mount the file systems:
```
mount /db2data/db2inst1/NODE0000
mount /db2data/db2inst1/NODE0001
mount /db2log/db2inst1/NODE0000
mount /db2log/db2inst1/NODE0001
```

Change ownership of the file system to the DB2 instance owner (user db2inst1 and group build in this example):
```
chown db2inst1:build /db2data/db2inst1/NODE0000
chown db2inst1:build /db2data/db2inst1/NODE0001
chown db2inst1:build /db2log/db2inst1/NODE0000
chown db2inst1:build /db2log/db2inst1/NODE0001
```
chown db2inst1:build /db2log/db2inst1/NODE0001

9. Now that the initial devices are set up on the Primary server, they need to be made known on the Backup server. The pairs must be split to be made available on the Backup server. From the Primary server issue the following commands:
   a. Split the pairs:
       HOMRCF:   
pairsplit –g db2dataN0
pairsplit –g db2logN0
pairsplit –g db2dataN1
pairsplit –g db2logN1
   
   HOARC:   
pairsplit –g db2dataN0 -rw
pairsplit –g db2logN0 -rw
pairsplit –g db2dataN1 -rw
pairsplit –g db2logN1 -rw
   
   b. Wait for pairsplit to finish:
       pairevtwait –g db2dataN0 -s psus -t 60
pairevtwait –g db2logN0 -s psus -t 60
pairevtwait –g db2dataN1 -s psus -t 60
pairevtwait –g db2logN1 -s psus -t 60

You can also check the status of the pairs using:
       pairdisplay –g db2dataN0
pairdisplay –g db2logN0
pairdisplay –g db2dataN1
pairdisplay –g db2logN1

10. Import volume groups from the Backup server:
    #smit vg
    ->Import a Volume Group

    Alternatively, issue the command:
    importvg –y vgdb2dataN0 hdisk4

    Import the remaining three volume groups from either smit or the command line:
    importvg –y vgdb2dataN1 hdisk5
importvg –y vgdb2logN0 hdisk6
importvg –y vgdb2logN1 hdisk7

11. Mount the file systems on the Backup server and ensure that the mount is successful:
    mount /db2data/db2inst1/NODE0000
    mount /db2data/db2inst1/NODE0001
    mount /db2log/db2inst1/NODE0000
    mount /db2log/db2inst1/NODE0001

12. Unmount the file systems on the Backup server:
    umount /db2data/db2inst1/NODE0000
    umount /db2data/db2inst1/NODE0001
    umount /db2log/db2inst1/NODE0000
    umount /db2log/db2inst1/NODE0001

13. Resynchronize the pairs from the Primary server and wait for the pairresync to complete:
    a. Issue the pairresync command:
       pairresync –g db2dataN0
       pairresync –g db2dataN1
       pairresync –g db2logN0
       pairresync –g db2logN1

    b. Wait for resync to finish:
       pairevtwait –g db2dataN0 -s pair -t 60
       pairevtwait –g db2dataN1 -s pair -t 60
       pairevtwait –g db2logN0 -s pair -t 60
       pairevtwait –g db2logN1 -s pair -t 60

       You can also check the status of the pairs using:
       pairdisplay –g db2dataN0
       pairdisplay –g db2dataN1
       pairdisplay –g db2logN0
       pairdisplay –g db2logN1

14. Install DB2 on both the Primary and Backup server if this is not already done. Consult the DB2 UDB documentation for more information on installing.

15. Create an identical EEE instance on both the Primary and Backup servers. The DB2 instance in our example consists of Partition 0 and Partition 1 on the same machine. The HOME file system is not part of a pair and may reside on an internal disk. Ensure that the UID/GID of the instance owner is identical across both machines. The current example uses db2inst1 as the instance owner. Since there are two separate copies of the instance directory, ensure that the changes are done on both instances, if required, when you are updating database manager configuration.
parameters and DB2 profile variables..

16. Start the db2inst1 instance on the Primary server:
   \textit{db2start}

17. Create a database in the db2inst1 instance on /db2data on the Primary server:
   \textit{db2 \textasciitilde create database test on /db2data}

18. With the test database created on the Primary server, update the NEWLOGPATH parameter for the Primary database to place the DB2 logs on the log file systems that were created:
   \textit{db2_all \textasciitilde db2 update db cfg for test using newlogpath /db2log/db2inst1}

\textbf{Note}: The new log path will not take effect until all users have disconnected from the database and the database has been deactivated. On the subsequent first connect to the database [or database activation], the active logs will be moved to the new log path. The \texttt{db2\_all} program is a program that will run a command at all DB2 partitions. Using the \texttt{\textbar|\textbar} option allows the command to be run in parallel at every DB2 partition.

The base configuration is now done. The following sections provide examples of how Hitachi split-mirroring technology and DB2 UDB suspend I/O technology can be utilized together.
Snapshot Database Scenario

This scenario starts with DB2 running on the Primary server and clients using the Primary database. The Backup server has DB2 stopped and the DB2 file systems for data and logs unmounted. This scenario assumes that the pairs are already defined and synchronized (Primary → Backup).

1. From the Primary server, issue the suspend command on the Primary database for each partition:

   Window A
   - `export DB2NODE=0`
   - `db2 “terminate”`
   - `db2 “connect to test”`
   - `db2 “set write suspend for database”`

   Window B
   - `export DB2NODE=1`
   - `db2 “terminate”`
   - `db2 “connect to test”`
   - `db2 “set write suspend for database”`

2. Split the pairs from the Primary server:

   HOMRCF:
   - `pairsplit –g db2dataN0`
   - `pairsplit –g db2logN0`
   - `pairsplit –g db2dataN1`
   - `pairsplit –g db2logN1`

   HOARC:
   - `pairsplit –g db2dataN0 -rw`
   - `pairsplit –g db2logN0 -rw`
   - `pairsplit –g db2dataN1 -rw`
   - `pairsplit –g db2logN1 -rw`

3. Wait for pairsplit to finish:

   - `pairevtwait –g db2dataN0 -s psus -t 60`
   - `pairevtwait –g db2logN0 -s psus -t 60`
   - `pairevtwait –g db2dataN1 -s psus -t 60`
   - `pairevtwait –g db2logN1 -s psus -t 60`

   You can also check the status of the pairs using:
   - `pairdisplay –g db2dataN0`
   - `pairdisplay –g db2logN0`
   - `pairdisplay –g db2dataN1`
   - `pairdisplay –g db2logN1`

4. Issue the resume command from the same connection in step 1 for each partition:

   Window A
   - `db2 “set write resume for database”`
Window B  

`db2 "set write resume for database"`

5. From the Backup server, mount the file systems:
   ```
   mount /db2data/db2inst1/NODE0000
   mount /db2log/db2inst1/NODE0000
   mount /db2data/db2inst1/NODE0001
   mount /db2log/db2inst1/NODE0001
   ```

6. Catalog the database test on the Backup server:
   ```
   db2 "catalog db test on /db2data"
   ```

7. Start the DB2 instance db2inst1 on the Backup server:
   ```
   db2start
   ```

8. The following command will perform crash recovery on each database partition and take it out of a suspended state:
   ```
   db2_all "||db2inidb test as snapshot"
   ```

The database on the Backup server is now in a transactionally consistent state at the time the suspend command was issued against the Primary database.

If a new snapshot image is needed, the DB2 instance on the Backup server should be stopped and the data and log file systems should be unmounted. Then the pairs should be resynchronized and the above steps performed again. Step 6 will not need to be rerun as the database has already been cataloged.

The above configuration can also be utilized as a standby system in the case of a failure on the Primary server. If the Primary server fails, the DB2 data and logs can be split so the Backup server has an image of the Primary database at the time the Primary system failed. Then the DB2 restart command can be applied to the Backup database to bring the database to a transactionally consistent point. For this scenario, the db2inidb command is not required, as the Primary database was never suspended.
Standby Database Scenario

This scenario starts with DB2 running on the Primary server. The Backup server has DB2 stopped and the DB2 file systems for data and logs unmounted. This scenario assumes that the pairs are already defined and synchronized (Primary → Backup).

Since the standby scenario requires logs from the Primary database to be applied to the Backup database, userexit should be enabled on the Primary server to facilitate the shipping of log files to the Backup database. Edit the DB2 userexit program and place it in the appropriate directory on both servers so that DB2 knows where the archive logs are located. The archive logs need to be shipped to the Backup server so the Backup database can apply these logs.

1. On the Primary server, enable the userexit program for each partition:
   ```
   db2_all “||db2 update db cfg for test using userexit on”
   ```

2. For the new userexit settings to take effect, all users must be forced off the database. The database will be placed into a backup-pending state, and a backup must be performed on each partition before any connections to the database are allowed:
   ```
   Window A
   export DB2NODE=0
   db2 “terminate”
   db2 “backup db test to /backup”
   
   Window B
   export DB2NODE=1
   db2 “terminate”
   db2 “backup db test to /backup”
   ```

   **Note:** The catalog partition must be backed up first.

3. Clients can now access the Primary database. Issue the suspend command on the Primary database for each partition:
   ```
   Window A
   db2 “connect to test”
   db2 “set write suspend for database”
   
   Windows B
   db2 “connect to test”
   db2 “set write suspend for database”
   ```

4. Split the data pairs from the Primary server:
   ```
   HOMRCF: pairsplit –g db2dataN0
           pairsplit –g db2dataN1
   
   HOARC: pairsplit –g db2dataN0 -rw
           pairsplit –g db2logN1 -rw
   ```

   **Note:** For this scenario the active logs on the Primary server are not split, and should
not be split and made available to the Backup server at this point.

5. Wait for Pair Split to finish:
   
   ```
pairevtwait –g db2dataN0 -s psus -t 60
pairevtwait –g db2dataN1 -s psus -t 60
   ```
   
   You can also check the status of the pairs using:
   
   ```
pairdisplay –g db2dataN0
pairdisplay –g db2dataN1
   ```

6. Issue the resume command from the same connection in step 3 for each partition:
   
   Window A   `db2 “set write resume for database”`
   
   Windows B  `db2 “set write resume for database”`

7. Mount the data file systems on the Backup server:
   
   ```
mount /db2data/db2inst1/NODE0000
mount /db2data/db2inst1/NODE0001
   ```

8. Catalog the database test on the Backup server:
   
   `db2 “catalog db test on /db2data”`

9. Start the DB2 instance db2inst1 on the Backup server:
   
   `db2start`

10. Place the Backup database into a rollforward pending state and take it out of a suspended state:
    
    `db2_all “|\|db2inidb test as standby”`

11. Establish the current time for the archived log files on the Primary server using the UNIX date command, and then force all the current logs to be archived. The session used to force the archived logs should not have a connection to the database.
    
    ```
date
db2 “archive log for db test on all nodes”
    ```

12. Copy archived logs on the Primary database to the retrieve path for the Backup database for each partition. Be careful not to roll forward through a log file on the Backup server that is only partially copied to the retrieve directory.

13. From the catalog partition on the Backup database, roll forward through the DB2 logs that have been archived from the Primary database. Roll forward to a point in time that is earlier than the time established in step 11:
    
    `db2 “rollforward database test to 2000-06-19- 19.56.00.0000”`

The following error may be encountered during the rollforward if the rollforward passes a point in time that is not available in the logs:
Roll-forward recovery on database "TEST" cannot reach the specified stop point (end-of-log or point-in-time) because of missing log file(s) on node(s) "0,1".

If this error occurs, the database cannot be currently taken out of a rollforward-pending state. To resolve this issue, you must copy over the next set of archived log files and ensure that you do not roll forward to a point past what is available in the archived logs.

**Note:** On the Backup server, the retrieved log files will be copied to /db2log/db2inst1/NODE0000 and /db2log/db2inst1/NODE0001 by DB2 during the rollforward, which is the current mount point for the log directories. Ensure that the instance owner has write access to this directory and that it is a sufficient size. When the log pair is split, and /db2log/db2inst1/NODE0000 and /db2log/db2inst1/NODE0001 are mounted, the contents of the mount points will no longer be visible. This does not cause a problem as any log files that may have been in this directory can still be retrieved from the retrieve path.

14. Continue with iterations of the previous rollforward commands until the Primary database fails, a database application logic error occurs, or the Backup database needs to be used.

If you do not need to roll forward through the most current logs from the Primary system, then proceed to step 19 and issue a rollforward to a point in time that is available in the archived logs.

15. At this point, ensure that all log files that have been archived on the Primary server are available to the Backup server.

16. Split the log pair from the Primary server:
   - **HOMRCF:**
     - pairsplit –g db2logN0
     - pairsplit –g db2logN1
   - **HOARC:**
     - pairsplit –g db2logN0 -rw
     - pairsplit –g db2logN1 -rw

17. Wait for pair split to finish:
   - pairevtwait –g db2logN0 -s psus -t 60
   - pairevtwait –g db2logN1 -s psus -t 60

   You can also check the status of the pairs using:
   - pairdisplay –g db2logN0
   - pairdisplay –g db2logN1
18. Mount the log file systems on the Backup server:
   
   mount /db2log/db2inst1/NODE0000
   mount /db2log/db2inst1/NODE0001

19. Issue a rollforward and stop command on the Backup database. If a database application logic error occurred, the user must roll forward to a point in time before the logic error occurred. Ensure that the point in time specified is available in the log files for each database partition. If the database needs to be recovered to the point when the Primary server failed, then a rollforward to the end of the logs should be performed:

   db2 "rollforward database test to end of logs and stop"
   
   or

   db2 "rollforward database test to 2000-06-19-20.57.00.0000 and stop"

The above scenario can also be run such that every rollforward is a rollforward to end of logs. If this method is used, the following error is expected when rolling forward through only the available archived logs:

   SQL4970N Roll-forward recovery on database "TEST" cannot reach the specified stop point (end-of-log or point-in-time) because of missing log file(s) on node(s) "0,1".

On the final rollforward to end of logs and stop, the active logs must be made available from the Primary server on failure of the Primary server.

Note: On the Backup server, you cannot do an end-of-log rollforward followed by a point-in-time rollforward.
Split-Mirror Online Backup Scenario

This scenario starts with DB2 running on the Primary server. The Backup server has DB2 stopped and the DB2 file systems for data and logs unmounted. This scenario assumes that the pairs are already defined and synchronized (Primary → Backup).

**Note:** This scenario is currently not supported for databases that contain SMS tablespaces.

1. Edit the DB2 userexit program and place it in the appropriate directory on the Primary server. Userexit should be enabled on the Primary server for each partition so that log files can be utilized for rollforward after restoring the backup image:
   
   ```
   db2_all "||db2 update db cfg for test using userexit on"
   ```

2. For the new userexit settings to take effect, all users must be forced off the Primary database. The database will be placed into a backup-pending state, and a backup must be performed on each partition before any connections to the database are allowed:

   **Window A**
   ```
   export DB2NODE=0
   db2 "terminate"
   db2 "backup db test to /backup"
   ```

   **Window B**
   ```
   export DB2NODE=1
   db2 "terminate"
   db2 "backup db test to /backup"
   ```

3. Clients can now access the Primary database. Issue the suspend command on the Primary database:

   **Windows A**
   ```
   db2 "connect to test"
   db2 "set write suspend for database"
   ```

   **Window B**
   ```
   db2 "connect to test"
   db2 "set write suspend for database"
   ```

4. Split the pairs from the Primary server:

   **HOMRCF:**
   ```
   pairsplit –g db2dataN0
   pairsplit –g db2logN0
   pairsplit –g db2dataN1
   pairsplit –g db2logN1
   ```

   **HOARC:**
   ```
   pairsplit –g db2dataN0 -rw
   pairsplit –g db2logN0 -rw
   pairsplit –g db2dataN1 -rw
   pairsplit –g db2logN1 -rw
   ```
Note: With DB2 UDB Version 7 FixPak 4 and later, you do not need the splitting of the logs for this scenario.

5. Wait for pair split to finish:
   
   `pairevtwait -g db2dataN0 -s psus -t 60`
   `pairevtwait -g db2logN0 -s psus -t 60`
   `pairevtwait -g db2dataN1 -s psus -t 60`
   `pairevtwait -g db2logN1 -s psus -t 60`

   You can also check the status of the pairs using:
   
   `pairdisplay -g db2dataN0`
   `pairdisplay -g db2logN0`
   `pairdisplay -g db2dataN1`
   `pairdisplay -g db2logN1`

6. Issue the resume command from the same connection in step 3 for each partition:
   
   Window A  
   `db2 "set write resume for database"`

   Window B  
   `db2 "set write resume for database"`

7. Mount the data and log file systems on the Backup server:
   
   `mount /db2data/db2inst1/NODE0000`
   `mount /db2log/db2inst1/NODE0000`
   `mount /db2data/db2inst1/NODE0001`
   `mount /db2log/db2inst1/NODE0001`

   Note: With DB2 UDB Version 7 FixPak 4 and later, you do not need the log file systems to be mounted.

8. Catalog the database test on the Backup server:
   
   `db2 "catalog db test on /db2data"`

9. Start the DB2 instance on the Backup server:
   
   `db2start`

10. The following command will place the Backup database into a rollforward-pending state and take it out of a suspended state for each partition:
    
    `db2_all "||db2inidb test as standby"`

11. Perform a DB2 backup on the Backup server for each partition:
    
    Windows A  
    `export DB2NODE=0`
    `db2 "terminate"`
    `db2 "backup database test to /splitbackup"`

    Windows B  
    `export DB2NODE=1`
    `db2 "terminate"`
12. Stop the DB2 instance on the Backup server:
   \texttt{db2stop}

13. Unmount the data and log file systems on the Backup server:
   \texttt{umount \textasciitilde/db2data/db2inst1/NODE0000}
   \texttt{umount \textasciitilde/db2log/db2inst1/NODE0000}
   \texttt{umount \textasciitilde/db2data/db2inst1/NODE0001}
   \texttt{umount \textasciitilde/db2log/db2inst1/NODE0001}

14. Resynchronize the pairs from the Primary server:
   \texttt{pairresync –g db2dataN0}
   \texttt{pairresync –g db2logN0}
   \texttt{pairresync –g db2dataN1}
   \texttt{pairresync –g db2logN1}

15. Wait for resync to finish:
   \texttt{pairevtwait –g db2dataN0 -s pair -t 60}
   \texttt{pairevtwait –g db2logN0 -s pair -t 60}
   \texttt{pairevtwait –g db2dataN1 -s pair -t 60}
   \texttt{pairevtwait –g db2logN1 -s pair -t 60}

   You can also check the status of the pairs using:
   \texttt{pairdisplay –g db2dataN0}
   \texttt{pairdisplay –g db2logN0}
   \texttt{pairdisplay –g db2dataN1}
   \texttt{pairdisplay –g db2logN1}

The DB2 backup image taken on the Backup server can then be restored to any DB2 server including the Primary server using the DB2 restore command.

1. Restore the DB2 backup to the desired machine for each partition:
   Window A \texttt{export DB2NODE=0}
   \texttt{db2 “terminate”}
   \texttt{db2 “restore database test from /splitbackup”}

   Window B \texttt{export DB2NODE=1}
   \texttt{db2 “terminate”}
   \texttt{db2 “restore database test from /splitbackup”}

2. Logs from the Primary server should be made available to the machine where the database restore was performed, thus allowing a rollforward through the logs.

3. After the restore completes, roll forward through the DB2 logs from the catalog partition. The rollforward command allows the option to roll forward to end of logs or to roll forward to a point in time. Ensure all logs that are required from the
Primary server are available to the new system:

$\text{db2 \ “rollforward database test to end of logs and stop”}$

or

$\text{db2 \ “rollforward database test to 2000-06-19-19.56.00.0000 and stop”}$
Split-Mirror Restore Scenario

This scenario starts with DB2 running on the Primary server. The Backup server has DB2 stopped and the DB2 file systems for data and logs unmounted. This scenario assumes that the pairs are already defined and synchronized (Primary → Backup).

1. Edit the DB2 userexit program and place it in the appropriate directory on the Primary server so DB2 knows where archived logs are located. Enable the DB2 userexit program for each partition:
   
   \[ \text{db2\_all} \ldots \text{db2 update db cfg for test using userexit on}\]

2. For the new userexit settings to take effect, all users must be forced off the database. The database will be placed into a backup-pending state, and a backup must be performed on each partition before any connections to the database are allowed:

   **Window A**
   
   \[ \text{export DB2NODE=0} \]
   \[ \text{db2 “terminate”} \]
   \[ \text{db2 “backup db test to /backup”} \]

   **Window B**
   
   \[ \text{export DB2NODE=1} \]
   \[ \text{db2 “terminate”} \]
   \[ \text{db2 “backup db test to /backup”} \]

3. Clients can now access the Primary database. Issue the suspend command on the Primary database for each partition:

   **Windows A**
   
   \[ \text{db2 “connect to test”} \]
   \[ \text{db2 “set write suspend for database”} \]

   **Window B**
   
   \[ \text{db2 “connect to test”} \]
   \[ \text{db2 “set write suspend for database”} \]

4. Split the data pair from the Primary server.

   **HORMRCF:**
   
   \[ \text{pairsplit –g db2dataN0} \]
   \[ \text{pairsplit –g db2dataN1} \]

   **HOARC:**
   
   \[ \text{pairsplit –g db2dataN0 -r} \]
   \[ \text{pairsplit –g db2dataN1 -r} \]

5. Wait for pair split to finish:

   \[ \text{pairevtwait –g db2dataN0 -s psus -t 60} \]
   \[ \text{pairevtwait –g db2dataN1 -s psus -t 60} \]

   You can also check the status of the pairs using:

   \[ \text{pairdisplay –g db2dataN0} \]
   \[ \text{pairdisplay –g db2dataN1} \]

6. Issue the resume command for each partition from the same connection in step 3:

   **Window A**
   
   \[ \text{db2 “set write resume for database”} \]
Window B  

```
    db2 “set write resume for database”
``` 

7. If the Primary database gets corrupted due to a database application error or some other error that prevents the database from being restarted, the data pair can be resynchronized and then a rollforward through the logs should be performed. At this point, DB2 should no longer be running on the Primary server.

8. Unmount the data file systems on the Primary server:
   ```
   umount /db2data/db2inst1/NODE0000
   umount /db2data/db2inst1/NODE0001
   ```

9. Resync the data pair from the Primary server so that the Primary copy contains the contents of the Secondary copy:
   - HOMRCF:
     ```
     pairresync -restore -g db2dataN0
     pairresync -restore -g db2dataN1
     ```
   - HOARC:
     ```
     pairresync -swapp -g db2dataN0
     pairresync -swapp -g db2dataN1
     ```

10. Wait for resync to finish:
    ```
        pairevtwait -g db2dataN0 -s pair -t 60
        pairevtwait -g db2dataN1 -s pair -t 60
    ```
    You can also check the status of the pairs.
    ```
    pairdisplay -g db2dataN0
    pairdisplay -g db2dataN1
    ```

11. If you are using HOARC, the prior pairresync in step 9 utilized the –swapp option, which also flips around the primary and secondary volumes. To flip this back, reissue the pairsplit and pairresync command one more time from the Primary server:
    ```
    pairsplit -g db2dataN0 -r
    pairsplit -g db2dataN1 -r
    ```
    Wait for the split to finish:
    ```
    pairdisplay -g db2dataN0
    pairdisplay -g db2dataN1
    ```
    Resync the pair, swapping the primary volume and secondary volume:
    ```
    pairresync -swaps -g db2dataN0
    pairresync -swaps -g db2dataN1
    ```
    Wait for the resync to finish:
    ```
    pairdisplay -g db2dataN0
    pairdisplay -g db2dataN1
    ```
12. Mount the data file systems on the Primary server:
   
   mount /db2data/db2inst1/NODE0000
   mount /db2data/db2inst1/NODE0001

13. Start the DB2 instance on the Primary server:
   
   db2start

14. On the Primary database issue the db2inidb command on each partition to take the
database image out of a suspended state and place it into a rollforward-pending state:
   
   db2_all "||db2inidb test as mirror"

15. On the Primary database from the catalog partition, roll forward though the DB2 logs
and stop. The rollforward command allows the option to roll forward to end of logs
or to rollforward to a point in time:
   
   db2 "rollforward database test to end of logs and stop"
   or
   db2 "rollforward database test to 2000-06-19-19.56.00.0000 and stop"
Incremental Restore (Pairresynchronization vs. Paircreate)

If more than 50% of the data has been changed since the last pairresync, it is usually faster to delete the pair (`pairsplit --g <group> --S`) and then recreate the pair. Pairresync updates the tracks that have been updated since the pair was last synchronized using a track update table. Paircreate creates a full mirror copy of the data without the overhead of referencing a track update table.
Appendix A – Splitting Pairs When the Primary Server is Down

Splitting ShadowImage Pairs

The examples in the preceding sections utilized a single CCI instance on each machine to aid in the readability of this paper. This configuration may not be ideal because if the Primary machine fails, one of the CCI instances will not be available, which will hinder successful execution of CCI commands. Thus, this appendix will provide an example of configuring two CCI instances on the Backup server with no CCI instances on the Primary server. Thus, all CCI commands must be executed from the Backup server.

Note: For more information on configuring CCI instances, please refer to the Hitachi Data Systems documentation.

1. Edit the /etc/services file on the Backup server and add the following information to the file so two CCI instances can communicate with each other:

   horcm0 11001/udp       #HORCM Instance 0 port
   horcm1 11002/udp       #HORCM Instance 1 port

   Note: The port number (xxxx/udp) used here is an example. To find a suitable port number, list the port numbers that are currently in use by viewing the /etc/services file. Choose any number that is not in use (the port number must be between 5000 and 65535).

2. Configure the horcm0.conf and horcm1.conf files on the Backup server.

   Backup system /etc/horcm0.conf

   HORCM_MON
   #ip_address     service     poll(10ms)     timeout(10ms)
machb           horcm0       1000           3000

   HORCM_CMD
   #dev_name     dev_name     dev_name
/dev/rhdisk28

   HORCM_DEV
   #dev_group     dev_name     port#     TargetID     LU#     MU#
db2dataN0      dataN0       CL1-E     0            0
/db2dataN1      dataN1       CL1-E     0            1
/db2logN0       logN0        CL1-E     0            2
/db2logN1       logN1        CL1-E     0            3

   HORCM_INST
   #dev_group     ip_address     service
db2dataN0      MACHB         horcm1
db2dataN1 machb horcm1
db2logN0 machb horcm1
db2logN1 machb horcm1

Backup system /etc/horcm1.conf

HORCM_MON
#ip_address service poll(10ms) timeout(10ms)
machb horcm1 1000 3000

HORCM_CMD
#dev_name dev_name dev_name
/dev/rhdisk28

HORCM_DEV
#dev_group dev_name port# TargetID LU# MU#
db2dataN0 dataN0 CL2-E 0 0
db2dataN1 dataN1 CL2-E 0 1
db2logN0 logN0 CL2-E 0 2
db2logN1 logN1 CL2-E 0 3

HORCM_INST
#dev_group ip_address service
db2dataN0 machb horcm0
db2dataN1 machb horcm0
db2logN0 machb horcm0
db2logN1 machb horcm0

3. Set the HORCC_MRCF environment variable to 1 so ShadowImage will be used:
   
   export HORCC_MRCF=1

4. Startup the CCI instances 0 and 1 on the Backup server:
   
   horcmstart.sh 0 1

5. Set the current CCI instance to 0:
   
   export HORCMINST=0

6. Create the ShadowImage pair:
   
   paircreate -g db2dataN0 -vl
   paircreate -g db2logN0 -vl
   paircreate -g db2dataN1 -vl
   paircreate -g db2logN1 -vl

   **Note:** The paircreate command is run from CCI instance 0 which corresponds to the CL1-E port on the Hitachi storage subsystem. Since machine MACHA is connected
to CL1-E, the primary volumes will be on MACHA.

This configuration requires all CCI commands to be run from the Backup server, and thus is not dependant on the Primary server being available.

**Splitting Remote Copy Pairs**

Remote Copy still requires a CCI instance to be running on each server. If the Primary machines fails, then issue the following commands from the Backup machine:

```
horctakeover -g db2dataN0 -t 30
horctakeover -g db2logN0 -t 30
horctakeover -g db2dataN1 -t 30
horctakeover -g db2logN1 -t 30
```

This will force the takeover allowing the most current data to be visible on the Backup machine.