IBM TotalStorage DS4000 Integrated Backup for Databases (IBD) with DB2

User Guide

Read Before Using

This product contains software that is licensed under written license agreements. Your use of such software is subject to the license agreements under which they are provided.

GA32-0474-00

IBM TotalStorage DS4000 Integrated Backup for Databases (with DB2)

User Guide

GA32-0474-00

First Edition (February 2005)

Contents

Preface

This user guide describes how to install, configure, and manage the backup and recovery of DB2 databases using the Integrated Backup for Databases (IBD) solution and the Tivoli Storage Manager (TSM) on IBM eServers and pSeries servers with databases stored on DS4000 storage subsystems.

Audience

IBD can be set up and managed by your IT staff. Installation and operation requires expertise in AIX, DB2, TSM, the DS4000 Storage Manager, HACMP (in cluster configurations), as well as experience with backup and recovery strategies.

IBM Business Partners offer a wide range of consulting services to help you migrate and consolidate your databases onto DS4000 storage subsystems and high performance pSeries servers. Data is often a company's most important asset. If you want help tailoring your data protection strategy or implementing IBD, contact your IBM account executive or your IBM Business Partner.

Getting Solution Software

IBD software can easily be downloaded from the IBM DS4000 Solutions and Premium Feature Web site. All that is required is information about the DS4000s that will participate in the backup solution and an IBD Solution Certificate for each DS4000 participating in the solution. IBD Solution Certificates can be purchased from IBM or your IBM Business Partner. See Chapter 2 of this user guide for more information.

Solution Support

IBM provides support for IBD. The IBD scripts are designed to run in any supported environment, without requiring modification. IBD script modification invalidates support entitlement.

Chapter 1. Introduction

This chapter provides a high-level description of Integrated Backup for Databases (IBD) and the business benefits of deploying IBD.

Policy-Based Backups

IBD is a policy-based backup solution for protecting data stored in DB2 databases. IBD, combined with the Tivoli Storage Manager (TSM), provides a complete end-to-end solution for backing up and restoring DB2 databases. The IBD solution uses DS4000 Copy Services to non-disruptively create point-in-time images of online databases. The backup process runs automatically and can be scheduled during off-peak hours to avoid impacting application performance. Because the process is fully automated, operator errors are avoided, backup integrity is enhanced, and system management costs are reduced.

IBD works with the TSM to provide a flexible range of database backup strategies for fast, reliable database recovery and long-term data protection. The IBD solution is easy to install and configure, and administrators can quickly add IBD support for any additional database servers and for any additional databases defined in a database server. IBD runs automatically on a scheduled basis, thereby reducing management costs. IBD was specifically designed and optimized for DB2 databases stored on DS4000 storage subsystems in:

- Direct connect topologies
- SAN topologies
- High availability cluster configurations

Technology Benefits

The key enabling technologies for IBD are designed into DS4000 subsystems to offload copy overhead from production servers. The FlashCopy feature creates point-in-time virtual database images and the VolumeCopy feature automatically transforms the FlashCopy images into physical database images that are available as Quick Recovery Volumes. This minimizes the time needed to recover damaged DB2 databases. IBD automatically invokes TSM for backing up your databases to disk or to tape for long-term data protection.

IBD runs transparently to users and applications while the databases storing their data are backed up, except for a momentary delay when a database to be backed up is write-suspended. The length of time that the database is write-suspended depends on its size and configuration. During this time, read I/Os are processed normally and write I/Os are queued to be run, but they are not run or acknowledged. Throughout the backup process, users and applications remain connected to DB2 and mission-critical applications continue to support business objectives around the clock.

Business Benefits

The e-business evolution makes continuous data availability a business imperative. In today's on-demand global economy, a policy-based plan for protecting data is crucial to your ongoing success. A comprehensive database protection strategy not only protects vital data but also eliminates the backup window, exploits the benefits of tiered storage, increases storage utilization, reduces server overhead, and enables fast, reliable recovery. Comprehensive policy-based automation is critical for business success.

- Continuous Operations: Split mirror online database backups run transparently to applications. With FlashCopy technology, point-in-time copies of data are nearly instantaneously created and then backed up. Competitiveness in a non-stop global economy requires information on-demand.
- Reliable Operations: Manual backups are inherently error prone, particularly when administrators are trapped between exponentially growing databases and shrinking backup windows, and you never know if you got the backup right until you need to recover a database. IBD enables the entire backup process to be carefully planned and automated; the system ensures that the backups are done right.
- Lower Cost Operations: The bottom line of continuous availability, error-resistant automated backup processes and policy-based data management is a sharper competitive edge in the market, more satisfied users, and a lower cost of operations.

References

IBM TotalStorage DS4000 Family and Storage Manager 9.10	http://www.redbooks.ibm.com/redpieces/abstracts/sg247010.html
IBM TotalStorage DS4000 Fibre Channel and Serial ATA Intermix Premium Feature Installation Overview	ftp://ftp.software.ibm.com/pc/pccbbs/pc_servers_pdf/gc26771300.pdf
Backing Up DB2 Using the Tivoli Storage Manager	http://publib-b.boulder.ibm.com/abstracts/sg246247.html?Open
IBM Tivoli Storage Manager Implementation Guide	http://publib-b.boulder.ibm.com/abstracts/sg245416.html?Open

Chapter 2. Licensing and Downloading Software

This chapter describes the process for licensing and downloading Integrated Backup for Databases (IBD) solution software.

IBD Solution Licensing

IBD is licensed on a “per subsystem” basis, like DS4000 Premium Features. One license is required for each DS4000 subsystem that participates in an IBD solution. An IBD “Solution Certificate” represents entitlement to download, install, and use IBD with a DS4000 subsystem. One Solution Certificate must be purchased from IBM or your IBM business partners for each DS4000 that will participate in your IBD solution. There are two solution certificate types: one for the DS4300 and another for the DS4400 and DS4500.

Downloading IBD Software

The process for obtaining IBD software is similar to the process for obtaining key files to enable Premium Features, such as FlashCopy, on your DS4000s.

Determine the number and type of DS4000 storage subsystems that will participate in your IBD solution and purchase a number of IBD Solution Certificates equal to the number of DS4000 storage subsystems.

Determine the serial numbers and SAFE-IDs of your DS4000s (the DS4000 Storage Manager generates a unique SAFE-ID for each DS4000 subsystem).

Access the IBM Solutions and Premium Features Web site: http://www-912.ibm.com/PremiumFeatures.

Enter your DS4000 serial numbers, IBD Solution Certificate numbers, and SAFE-IDs, and then download IBD solution software.

Chapter 3. IBD Solution Overview

This chapter describes the concepts and technologies used in the Integrated Backup for Databases (IBD) solution, and provides an overview of the IBD processes. The backup and restore processes are described in Chapter 8 and Chapter 9.

Solution Concepts

IBD leverages the DB2 split mirror backup technology and DS4000 Copy Services to non-disruptively back up databases and create Quick Recovery Volumes for fast database restores.

Split Mirror Backup

The DB2 **set write suspend** and **set write resume** commands enable backup of online databases using the split mirror technique. During the split mirror process, users and applications remain connected to their databases. Creating a database mirror and mapping it to a recovery server for backup avoids the problems and downtime of disconnecting and reconnecting users and minimizes overhead on production systems. IBD uses these DB2 commands and DS4000 Copy Services to non-disruptively back up online databases.

DS4000 Copy Services

IBD uses two Copy Services: FlashCopy and VolumeCopy. FlashCopy creates virtual copies of data as the data existed at a particular point-in-time. VolumeCopy creates physical images of logical or virtual volumes. FlashCopy and VolumeCopy are combined in IBD to create virtual and physical mirrors. The mirrors are mounted on a recovery server and are backed up without imposing overhead on production servers.

Operation:	Creates:	Which Are:
FlashCopy	FCvolumes	Virtual point-in-time images of database volumes.
VolumeCopy	VCvolumes	Physical point-in-time image of database volumes.

In the IBD solution, the volumes created by FlashCopy and VolumeCopy operations are referred to as FCvolumes and VCvolumes, respectively.

Quick Recovery Volumes (QRV)

The VCvolumes can be used as “Quick Recovery Volumes” to accelerate the process of recovering damaged DB2 databases. Restoring a corrupt DB2 database by creating, mapping and mounting new volumes, and then copying data from TSM disk-based storage pools or tape media is time consuming. With IBD, a corrupt database can be quickly dismounted, the corresponding VCvolumes can be mounted in its place, it can be initialized as a DB2 database, and rolled forward with minimum application downtime.

Automation

IBD combines FlashCopy and VolumeCopy technologies with the automation necessary to avoid cockpit errors, enforce database protection policies, reduce system management costs and increase business efficiency. The IBD backup process is fully automated and integrated with TSM making it easy-to-deploy, easy-to-use and easy-to-maintain as databases expand or are reconfigured.

Server Associations

IBD uses the notion of an association between one or more DB2 servers and a recovery server. The association defines a relationship between a set of DB2 instances running on database servers and a recovery server that provides backup services to the databases in the DB2 instances (see Figure 1).

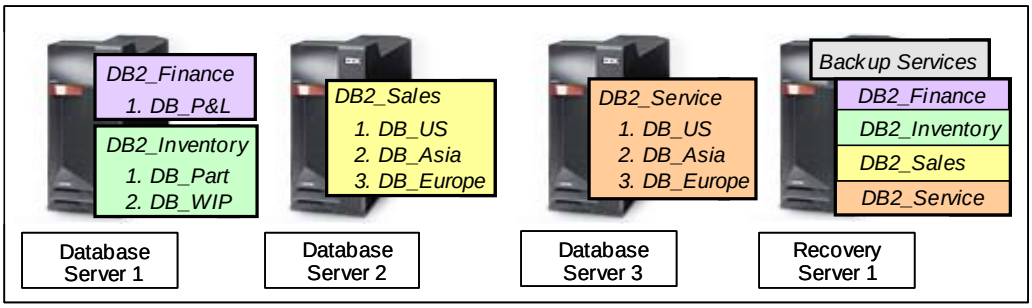


Figure 1. Association between database servers and a recovery server

IBD creates a backup set that includes table spaces, database paths, any other files that you want to include as part of the backup, and an IBD-generated metadata file for each database backup. The metadata file is used to facilitate database recovery to the same physical server that previously managed it, or to any available server.

Self-Installing “Black Box” Scripts

An IBD tar file is installed on a recovery server and then the self-extracting installer installs IBD on every DB2 servers participating in the solution.

Backup solutions can be difficult to maintain as databases are added or reconfigured if the scripts need to be modified pursuant to database changes. The IBD Configuration File provides a single administrative interface in which backup parameters are specified and can be easily updated to reflect any database changes.

Configuration File: A central administrative interface to IBD.

Automation scripts: Automate the backup process and never require modification.

The Configuration File defines the backup strategy and is stored on the recovery server (see Figure 2). A single file can be used to back up one or all of the databases managed by a DB2 instance. If certain databases need to be backed up at one time (for example, 8 a.m.), and other databases at another time (for example, 4 p.m.), or if different backup policies need to be applied to different databases, then multiple configuration files can be easily created to accommodate these requirements.

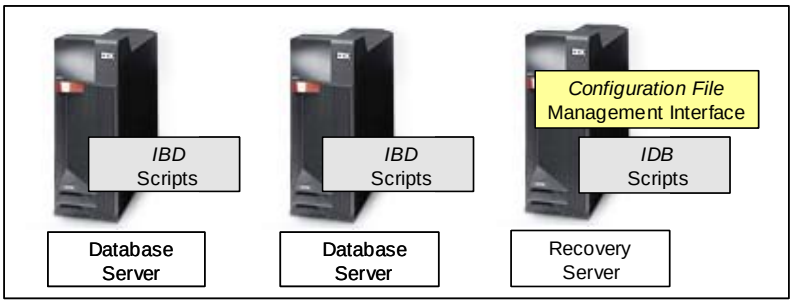


Figure 2. IBD Solution components

IBD Backup Cycle

IBD iteratively backs up the databases specified in a Configuration File. If multiple databases are specified, they are sequentially backed up in cycles (see Figure 3).

During each backup cycle, the action of writing data to the database that has been identified for backup is momentarily suspended (read I/Os are processed normally but write I/Os are queued and are not run or acknowledged until IBD issues the **set write resume** DB2 command). The database is then FlashCopied (creating FCvolumes) and writing to the database is resumed. The backup process then follows one of two paths:

- VolumeCopy the FCvolumes (which creates physical on-disk database images called VCvolumes) and, optionally, TSM is called to back up the VCvolumes; otherwise it goes to the next backup cycle (if other databases are queued for backup), or
- TSM is called to back up the FCvolumes and then it goes to the next backup cycle (if other databases are queued for backup).

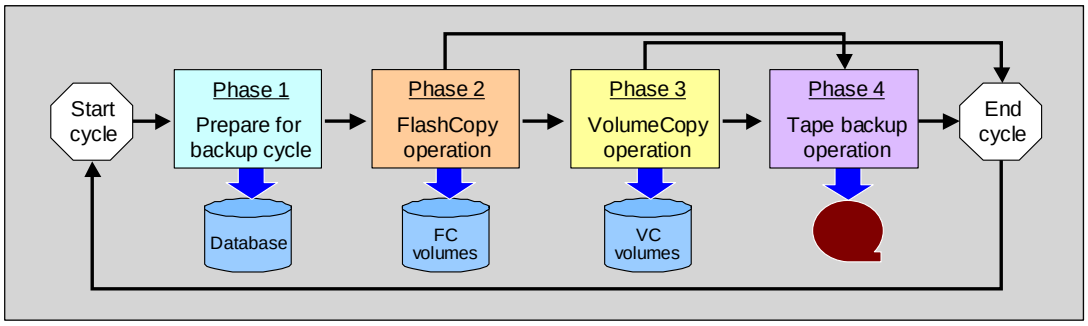


Figure 3. IBD backup cycle

Policy-Based Backups

IBD uses free space on a DS4000 for FCvolumes and, optionally, VCvolumes. The initial IBD storage configuration is shown in Figure 4.

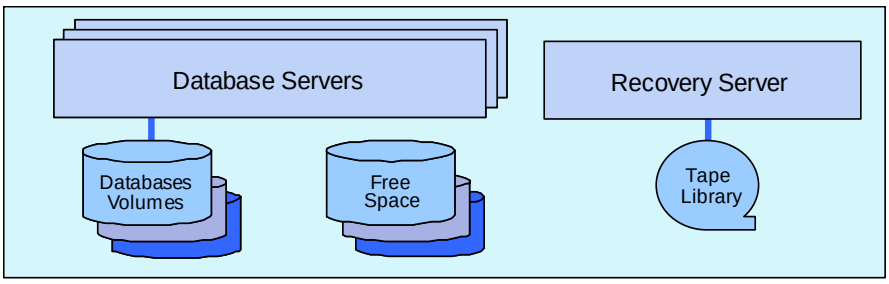


Figure 4. IBD uses free space on a DS4000

Capacity is allocated from free space for the FCvolumes and, optionally, to VCvolumes when IBD runs (see Figure 5).

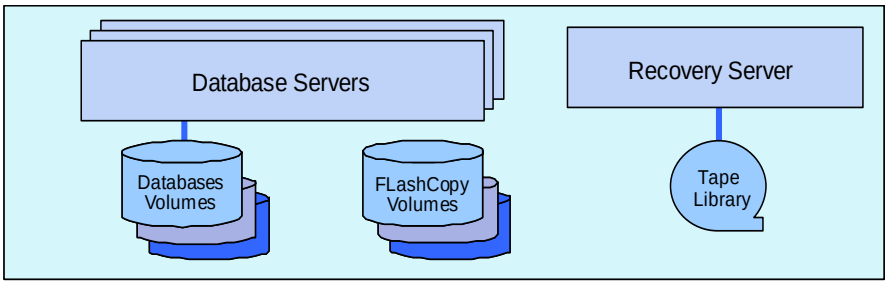


Figure 5. Free space is pre-allocated for FCvolumes

When a backup operation runs, IBD reads and validates the Configuration File and automatically creates FCvolumes and, optionally, VCvolumes. IBD then runs the DB2 **set write suspend** command to temporarily suspend writes to the database that has been identified for backup. IBD invokes FlashCopy and, immediately after the FlashCopy data structures are set up, IBD issues the DB2 **set write resume** command, which allows DB2 to resume write I/O activity to the database (see Figure 6). IBD backs up the databases but does not back up the related database log files because the consistency of an archive log FlashCopy cannot be guaranteed (the **set write suspend** command does not prevent active logs from being written to the archive logs during a FlashCopy operation). Therefore, the administrator must back up the DB2 log files to a safe location using TSM.

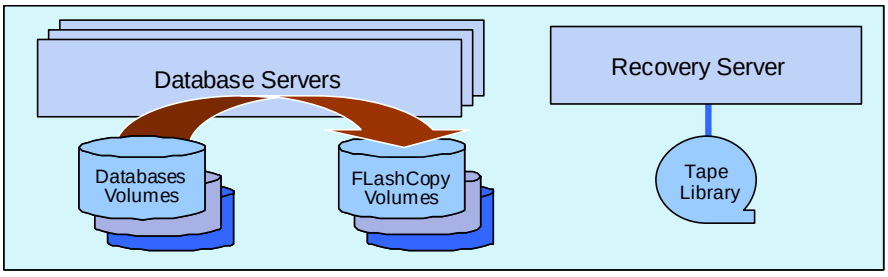


Figure 6. IBD creates virtual database images

If the backup policy for this database is to offload the FlashCopy volumes using TSM, IBD mounts the FlashCopy volumes on the recovery server and calls TSM (see Figure 7).

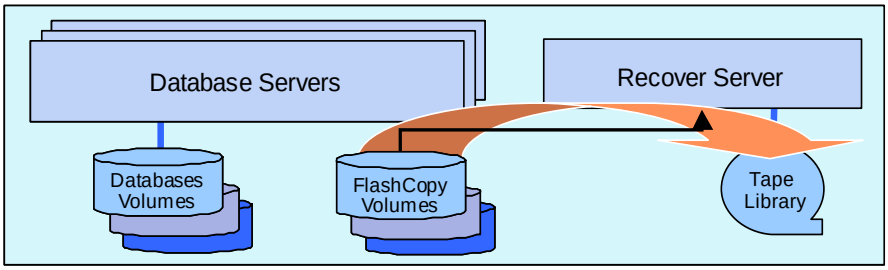


Figure 7. IBD calls TSM to back up the virtual DB image

If the backup policy calls for creating Quick Recovery Volumes, IBD automatically calls VolumeCopy and initiates a VolumeCopy operation (see Figure 8). Because FlashCopy and VolumeCopy run at the subsystem level, copy overhead is offloaded from the database production servers.

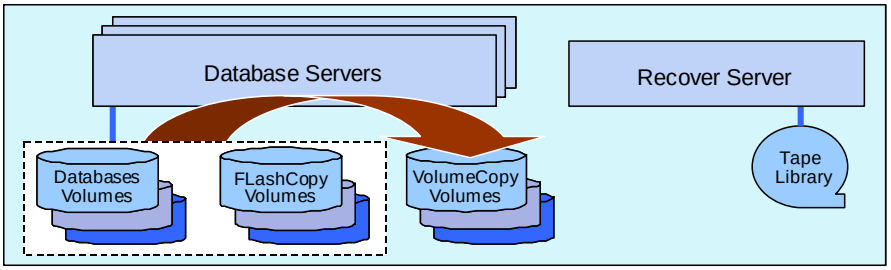


Figure 8. IBD optionally creates physical DB images

When the VolumeCopy operation completes, IBD maps the VCvolumes to the recovery server and, optionally, calls TSM to offload the VCvolumes for long-term database protection (see Figure 9).

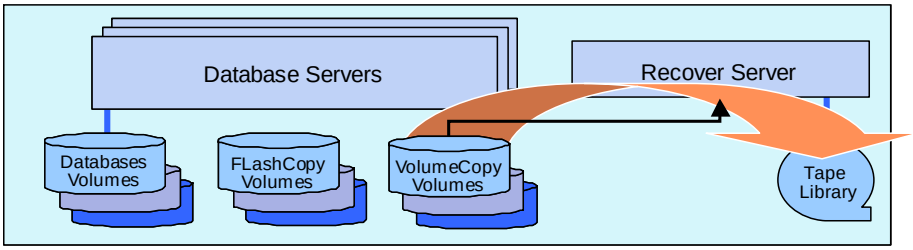


Figure 9. Long-term database protection

Fast Restores

If a database becomes corrupted, the VCvolumes can be initialized as a database for quick recovery. The damaged database volumes are dismounted, the corresponding VCvolumes are mounted in a roll-forward pending state, and the logs that have been generated since the VCvolumes were created are applied (see Figure 10).

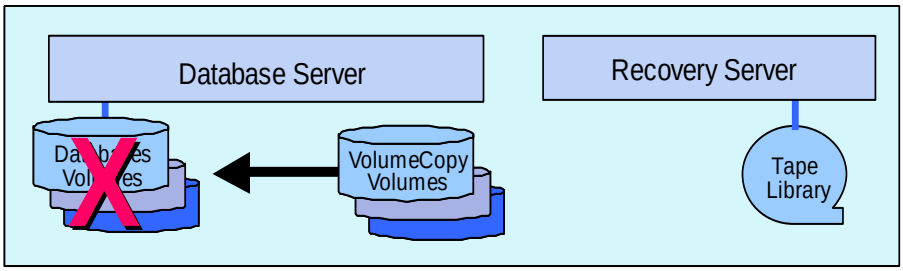


Figure 10. Fast restore of a damaged database

Chapter 4. Backup Planning

This chapter describes the elements of a comprehensive database protection strategy, explains FlashCopy and VolumeCopy operations, and defines database – recovery server associations.

Database Protection Strategy

Planning your database protection strategy is the most important area for consideration before implementing an IBD solution. You should consider the value to your organization of the data in each database, as well as the costs to your company if a database became unavailable. For each database, your DBA and storage administrator need to determine whether a quick recovery strategy, such as creating VCvolumes, is cost-justified (because additional disks are required to implement it) or if writing the FCvolumes to tape is sufficient.

A data backup strategy is only one aspect of an overall data protection strategy. Your system should be designed so that it is fault-tolerant from top to bottom by using clustered servers, a fault-tolerant SAN with redundant I/O path between each server and the storage subsystems, and fault-tolerant RAID configurations. Your DS4000 includes redundant I/O ports, controllers, power supplies, disk connections, and sophisticated firmware to prevent single points of failure from disrupting operations. None of these precautions can individually guarantee continuous availability of your databases but, in combination, they minimize the probability of a system outage and mitigate the impact of a failing system component.

Your backup strategy should define the requirements your backup solution must satisfy, such as:

- Types of events that could cause loss of database access.
- Required speed of recovery from a failure.
- Profile of daily database activity, to determine when to back up databases.
- Recovery points (the point-in-time you need to recover to).
- Units of recovery (databases, logs, control and other files).

IBD provides the flexibility to design a data protection system that includes quick recovery from recent events, as well as recovery from events that might be days, or even weeks, old.

Planning for FlashCopy and VolumeCopy

In IBD, a profile of daily database activity is important as any database copy activity (Mirrors, FlashCopies, VolumeCopies, or tape-based copies) imposes unavoidable overhead. With IBD, overhead is overloaded from production servers and contained within the storage subsystem and the recovery server, in contrast to host-based software solutions that add significant overhead to production servers and the SAN fabric.

DS4000 Copy Services are powerful data replication technologies that efficiently add layers of database protection. FlashCopy and VolumeCopy are local (or intra-site) Copy Service capabilities. IBD synergistically uses FlashCopy and VolumeCopy to near-instantaneously create virtual database copies and transforms them into physical database images.

DS4000 storage subsystems also provide two remote (or inter-site) Copy Service features for disaster recovery: Synchronous Metro Mirrors, and asynchronous Global Mirrors. In this release, IBD does not automatically use these remote Copy Service features.

FlashCopy and FCvolumes

IBD uses FlashCopy technology to create near-instantaneous virtual copies of database volumes as they existed when IBD issues the **write suspend** command.

When FlashCopy is invoked, a copy-on-write relationship is established between source database volumes and target FCvolumes. Once the relationship is set-up, the target volumes immediately look exactly like the source volumes as of the point-in-time the FlashCopy operation was invoked. Control is immediately returned to the operating system and the target FCvolumes are available for read/write access.

Initially, the target FCvolumes are essentially “empty.” While the relationship exists between source database volumes and target FCvolumes, a background task tracks changes to the database volumes and copies “before state” data to the FCvolumes as database updates occur. A FlashCopy data structure keeps track of the data blocks that have been copied from database source volumes to FCvolumes.

Only one FlashCopy relationship between source and target volumes can exist at any point in time and the relationship remains in effect until it is terminated. As the FCvolumes age, copy activity to the target volumes generally declines because the “before state” of disk blocks only needs to be copied once to the FCvolumes. If an application wants to read a block from an FCvolume that has not yet been copied, the data is read from the source database volume; otherwise, the read is satisfied from the FCvolume.

The IBD default capacity setting for FCvolumes is 20% of the capacity of the associated source database volumes. As the data stored on FCvolumes approaches the capacity limit of FCvolumes, warning messages are sent to the administrator and the capacity of the FCvolumes can be dynamically increased without interrupting the FlashCopy process using the DS4000 Storage Manager. If the warning messages are ignored and the data stored on the FCvolumes equals the capacity limit of the FCvolumes, the next database I/O that requires “before state” data to be copied to the FCvolumes will cause the FlashCopy process and IBD to fail. This situation is remedied by incrementing the FCvolume capacity parameter in the IBD Configuration File and re-running the backup job. When planning your DS4000 configuration, it is important to consider the capacity required for the FCvolumes.

Because FlashCopy operates at the subsystem level, it can only be used to copy source data stored on the same DS4000 subsystem. Therefore, the FCvolumes must be attached to the same DS4000 as the associated source database volumes. The most efficient way to backup database control files related to a database is to move these files to the DS4000 if they are presently stored on a server’s internal disks.

VolumeCopy and VCvolumes

VolumeCopy creates physical images of logical or virtual volumes. When VolumeCopy is invoked, a relationship is setup between source and target volumes, and data is copied block-by-block from source to target volumes. Consequently, VCvolumes must have the same or a greater capacity than the associated source volumes. IBD optionally uses VolumeCopy to create physical images of FCvolumes as of the point-in-time when the FCvolumes were created. Because FCvolumes are virtual copies of database volumes, the VCvolumes must have the same or a greater capacity than the source database volumes in the relationship with the FCvolumes.

Database Volumes, FCvolumes, and VCvolumes

After IBD creates FCvolumes, it automatically invokes VolumeCopy (if the VolumeCopy parameter in the IBD Configuration File is enabled) and creates VCvolumes. FlashCopy and VolumeCopy operations run as background tasks on the DS4000, in parallel with application I/O to databases. Because FlashCopy and VolumeCopy operations impose overhead on subsystem resources, database backups should be planned for off-peak periods of database activity.

A one-to-one-to-one relationship exists between database source volumes, FCvolumes and VCvolumes, as shown in Figure 11.

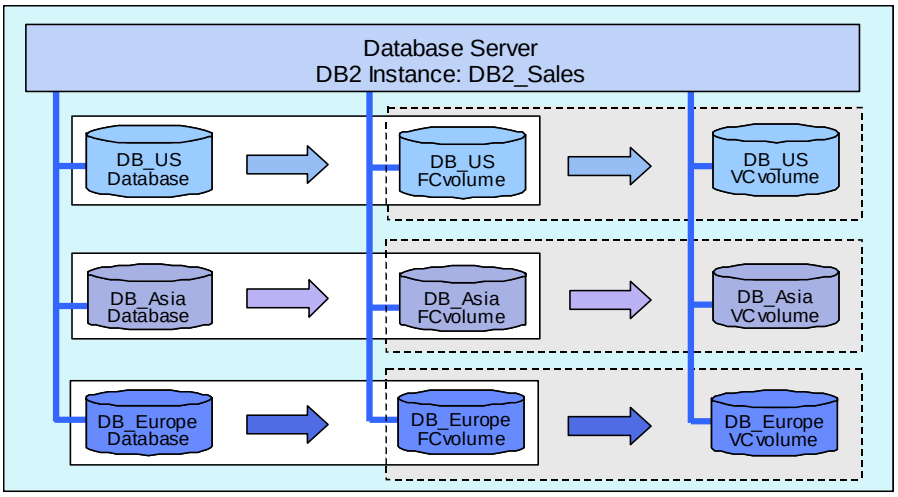


Figure 11. Database volumes, FCvolumes and VCvolumes

Database and Recovery Server Associations

IBD implements the notion of a “Database – Recovery Server Association,” which is a relationship between one or more database servers and a recovery server that provides backup services to the associated database servers. Figure 12 illustrates a dual-server association.

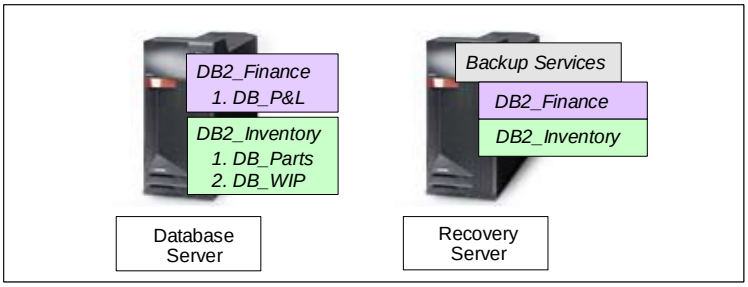


Figure 12. Database -- recovery server association

Figure 13 shows as association involving multiple database servers. In this case, the three database servers will be backed up by recovery server 1.

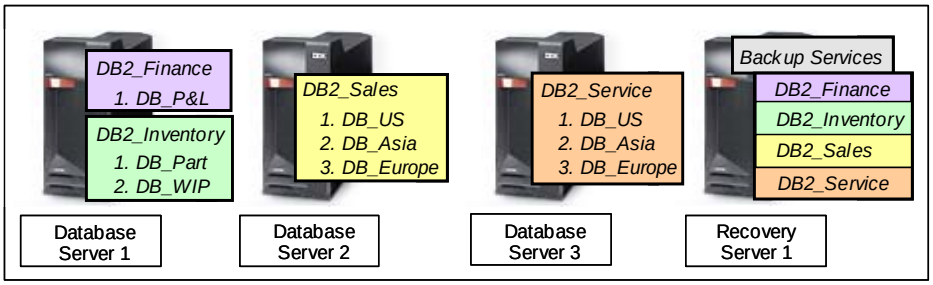


Figure 13. Association with multiple database servers

Figure 14 shows two association; the databases in the DB2_Finance and DB2_Inventory instances are backed up by recovery server 1, and the databases in the DB2_Sales and DB2_Services instances are backed up by recovery server 2. With multiple associations, databases can be backed up in parallel.

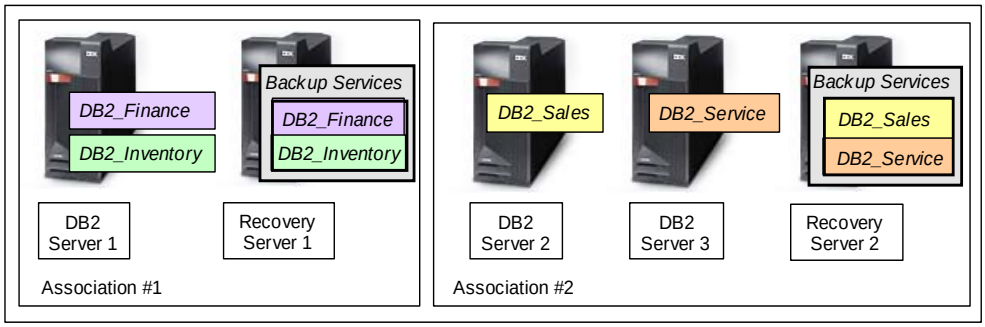


Figure 14. Two database – recovery associations

Chapter 5. System and DS4000 Configuration

This chapter describes the Integrated Backup for Databases (IBD) server, system software, and storage requirements and restrictions.

Hardware Server Planning

IBD is intended for low-end and mid-range DB2 environments and is supported on eSeries and pSeries servers running the AIX operating system.

Minimum Server Configuration

The IBD architecture is premised on the presence of one or more database servers and one recovery server with DB2 running on the database servers and a TSM server on the recovery server. Refer to http://www.tivoli.com/support/storage_mgr/tivolimain.html for information on configuring TSM.

Note: TSM clients are required on all servers in the association.

The minimum supported IDB environment includes two hardware servers. The servers in an IBD environment can host other applications; IBD does not impose any restrictions regarding other applications running on the database or recovery servers.

Maximum Server Configuration

IBD does not impose any restriction on the maximum number of servers in an IBD configuration. IBD has been tested with up to four pSeries servers.

Supported Servers

IBD is supported on the following low-end and mid-range IBM servers:

- eServer p5, models 520, 520 Express, 550, and 570
- pSeries models 615, 630, 650, 655, and 670

Supported Hardware

All hardware components (and their firmware revision levels), such as Host Bus Adapters (HBA), switches, storage and tape subsystems must be IBM approved components (see the IBM Hardware Compatibility List on the IBM Web site for further information).

System Software Planning

This section covers the database server and recovery server software requirements.

Database Server Software Requirements

The required and optional software on all database servers is as follows:

- AIX V5.2, Maintenance Level 5
- Journaling File System (JFS and JFS 2), 32 and 64 bit versions
- AIX Volume Manager, V5.2.0.30

- AIX Failover Driver, V5.2.0.50
- DB2 UDB, V8.2
- Tivoli Storage Manager, V5.2 (client only)
- DS4000 Storage Manager, V9.1 (optional)
- DS4000 Storage Manager's SMutils and SMcli, V9.1
- Sendmail

Note: The DS4000 Storage Manager is required to configure the storage for IBD; it can be installed on any server with access to the subsystems used with IBD.

SMutils are DS4000 Storage Manager utilities (SMdevices and hot_add are used by IBD). See the DS4000 Storage Manager section in this chapter for more information.

Recovery Server Software Requirements

The required and optional software on the recovery server is as follows:

- AIX V5.2, Maintenance Level 5
- Journaling File System (JFS and JFS 2), 32 and 64 bit versions
- AIX Volume Manager, V5.2.0.30
- AIX Failover Driver, V5.2.0.50
- Tivoli Storage Manager, V5.2 (client and server)
- DS4000 Storage Manager, V9.1 (optional)
- DS4000 Storage Manager's SMutils and SMcli utilities, V9.1
- Sendmail

System Support and Restrictions

The following restrictions must be taken into consideration:

- IBD requires a minimum of two hardware servers.
- Supported eServers servers – p5, models 520, 520 Express, 550 and 570.
- Supported pSeries servers – 615, 630, 650, 655 and 670.
- All system components, such as HBAs, must be listed on the IBM Hardware Compatibility List.
- Partitioned DB2 databases (logical and physical partitions) are not supported.
- AIX, V5.2, Maintenance Level 5, is the only supported operating system.
- AIX Journaling File System and Enhanced JFS (or JFS 2) are the only supported file systems.
- Filesystems, and their logs, that are used to store DB2 objects must be used exclusively for storing DB2 objects, and must be stored in the same Volume group.
- The active and archive logs cannot reside in the same Volume group as the corresponding database and its table spaces.
- The archive logs have to be backed up separately using TSM. IBD will not back up the archive logs.
- While a restore operation is in progress, IBD cannot be used to back up the database being restored until the restore operation completes.

HA Cluster Restrictions

The database servers and the recovery server can be configured in a HACMP cluster.

- IBD-protected database servers can failover to other database servers, but not to the recovery server used in an IBD association. The IBD design does not allow a DB2 server to be hosted on the same physical machine that hosts the IBD scripts that are installed on the recovery server.
- If the recovery server fails, IBD will also fail (it is not an HACMP-aware application and will not failover to another server).
- IBD does not support physical database partitions, such as in environments where a database is partitioned across cluster nodes.

Configuring DS4000 Subsystems

This section describes the processes used in determining requirements and configuring the DS4000 subsystems to work with IBD.

Determining the Optimal FlashCopy (FCvolume) Capacity Setting

IBD creates an FCvolume for the databases that it backs up. Because FCvolume capacity is reused in each backup cycle, you do not need to allocate capacity for an FCvolume for each database that you back up. Instead, you need only to allocate FCvolume capacity based on the largest database that you back up with IBD. For example, if you plan to use IBD to back up databases “A” and “B,” allocate FCvolume capacity for the larger of the two databases. Refer to the example in the “DS4000 Capacity Planning Example” section in this chapter.

If your backup policy for a particular database calls for creating a VCvolume, the FCvolume will be maintained until the VCvolume is created and then disabled. The adequacy of the 20% default threshold capacity setting (Configuration File) for the FCvolume depends primarily on the priority level assigned to the VOLUME_COPY_PRIORITY parameter (Configuration File). If the priority level for VCvolume creation is high or highest, the 20% default setting should be adequate in most environments. At lower priority level settings, it might be necessary to increase the FCvolume capacity setting.

If your backup policy for a particular database is to offload the FCvolume to tape (in other words, your policy does not include creating a VCvolume), the critical factor for the FCvolume threshold capacity setting is the time required to copy the FCvolume to tape because the FCvolume will be maintained until the tape backup operation completes. In this case, the adequacy of the 20% default capacity setting for the FCvolume depends primarily on the performance characteristics of your tape device and the available bandwidth between the FCvolume and the tape device.

For either backup policy, some experimentation with the capacity threshold setting for an FCvolume might be necessary in order to determine the optimal setting. FlashCopy issues warnings as its capacity threshold is approached and fails if the subsystem attempts to copy more data to it than is allowed by the FCvolume capacity setting. If this occurs, increment the FLASH_REPOSITORY_PERCENT_OF_BASE parameter in the Configuration File. We recommend increasing this value in 10% increments to determine the optimal setting.

For each database that you plan to back up with IBD, determine:

- The backup policy including VOLUME_COPY_PRIORITY setting (if appropriate).
- The backup cycle.

Set up a Configuration File and conduct a test run using the IBD default setting for the FLASH_REPOSITORY_PERCENT_OF_BASE= parameter to determine if the default setting is adequate. Because database activity varies during the course of a day, run the test at the same time of day as indicated in your backup cycle plan. If a backup cycle fails (Error Code 77), this probably indicates that the threshold capacity setting for the FCvolume is inadequate. The following actions can be taken individually or in combination (if VolumeCopy is enabled) to resolve the problem:

1. Increase the threshold capacity setting for the FCvolume (increment the FLASH_REPOSITORY_PERCENT_OF_BASE parameter in the Configuration File). The implication of this action is that additional disk capacity will be allocated for FCvolumes. Increment this parameter in 10 percentage point increments, for example, 20% to 30%.
2. If your backup policy includes creating a VCvolume, increase the VolumeCopy priority setting (change the VOLUME_COPY_PRIORITY parameter setting to a higher priority level, such as from Medium to High or Highest). The implications of this action are: a) VolumeCopy operation will be completed faster, b) Because the VolumeCopy operation completes faster, the length of time the FCvolume needs to be maintained will decrease and, therefore, the FCvolume requires less space, and c) Because more controller resources are allocated to the VolumeCopy operation, less resources will be available to service host I/O's and this could impact application performance.
3. If your backup policy includes creating a VCvolume, you can allocate VCvolume capacity from higher performance disks or spread the VCvolume across more disks. Either action will probably decrease the time to create the VCvolume and the time the FCvolumes are maintained.

Note: Because the required FCvolume capacity is a function of database size and workload, we recommend that you add buffer capacity to the FCvolumes particularly if you expect the size of the database or the database workload to increase over time.

DS4000 Capacity Planning

IBD supports tiered storage; mixing Fibre Channel and SATA disks to optimize across cost and performance. The storage capacity required to implement an IBD solution is the sum of the storage capacities required for your databases including database logs plus FCvolumes and VCvolumes.

The backup process creates FCvolumes and optionally, VCvolumes. IBD maintains two sets of VCvolumes (for the last two backup cycles) and reuses them in a round-robin fashion. To determine the total storage capacity required to implement your IBD backup solution, you must first define the backup policy for each database as being either:

- FlashCopy and offload to tape
- FlashCopy and VolumeCopy (and optionally, offload to tape)

The FCvolume capacity for a particular database depends on the length of time the FCvolume is maintained. The storage capacity for databases using a FlashCopy then off-load to tape backup policy is the sum of the capacities required for:

- Database volumes
- FCvolumes (default setting is 20%)

The storage capacity for databases using a FlashCopy then VolumeCopy backup policy is the sum of the capacities required for:

- Database volumes
- FCvolumes (default setting is 20%)
- VCvolumes (2x the storage capacity of the database volumes)

The total storage capacity required for your IBD solution is the aggregate capacity of all database volumes including logs, FCvolumes (based on the largest database), and VCvolumes.

Note: Because capacity allocated for FCvolumes is reused on each backup cycle, you do not need to allocate FCvolume capacity for each database, only for the database that will require the largest FCvolume capacity. See the example in the next section.

DS4000 Capacity Planning Example

To clarify storage requirements an example is provided with two scenarios. Because capacity planning is primarily a function of the number and size of your databases, and your backup policy for each database, the example ignores details such as the number of DB2 instances, backup schedules and so on.

Table 1 summarizes the backup policy assumptions. Databases “A” and “B” are critical to the operation of a business and require a backup policy with the fastest possible database recovery time. Databases “C” and “D” are less critical and a slower database recovery time will not significantly affect business operations.

Table 1. Example: Backup policies for each database

Database	Capacity	Backup Policy
Database A	100 GB	FlashCopy → VolumeCopy
Database B	200 GB	FlashCopy → VolumeCopy
Database C	300 GB	FlashCopy → Tape
Database D	400 GB	FlashCopy → Tape

Scenario 1:

In scenario 1, assume that the 20% default threshold capacity setting for the FCvolumes is adequate. The total storage capacity required to implement an IBD solution for this scenario is 1680 GB; 1000 GB for the production DB2 databases and logs, 80 GB for the FCvolumes and 600 GB for the VCvolumes.

Table 2. Example: Using the default capacity setting for FCvolumes.

Database	Database Capacity	FCvolume Capacity Setting	FCvolume Capacity	VCvolume Capacity
Database A	100 GB	20%	20 GB	200 GB
Database B	200 GB	20%	40 GB	400 GB
Database C	300 GB	20%	60 GB	--
Database D	400 GB	20%	80 GB	--
All Databases	1000 GB		80 GB	600 GB

Scenario 2:

Scenario 2 uses the same set of backup policies from Table 1, but assumes that more FCvolume capacity is required for database C because the performance of the tape device requires the corresponding FCvolumes to be maintained for a relatively long period of time and also assumes that because database D is used for data mining, its I/O profile is dominated by read I/Os (whereas database C has a more even distribution of reads and writes) and it requires less FCvolume capacity than the default setting. Using these assumptions, the total storage required to implement an IBD solution is 1720 GB: 1000 GB for the production DB2 databases and logs (unchanged), 120 GB for the FCvolumes (an increase of 40 GB) and 600 GB for the VCvolumes (unchanged) as shown in Table 3.

Table 3. Example: Modifying the default FCvolume capacity setting

Database	Database Capacity	FCvolume Capacity Setting	FCvolume Capacity	VCvolume Capacity
Database A	100 GB	20%	20 GB	200 GB
Database B	200 GB	20%	40 GB	400 GB
Database C	300 GB	40%	120 GB	--
Database D	400 GB	10%	40 GB	--
All Databases	1000 GB		120 GB	600 GB

DS4000 Storage Manager

The Storage Manager provides a powerful yet easy to use management interface. All storage administrative tasks, such as, configuration, expansion, maintenance and performance tuning can be performed without interrupting system I/O. IBD uses two DS4000 utilities.

hot_add Utility

The Storage Manager includes the hot_add utility. This host-based utility is used by IBD to register volumes with the AIX operating system. Once volumes have been created and the volume-to-LUN mappings defined, IBD runs the hot_add utility to ensure that the operating system is aware of any newly created volumes.

SMdevices Utility

The Storage Manager includes another utility called SMdevices. This host-based utility is used by IBD to associate physical device names (AIX perspective) with volume names (DS4000's perspective).

DS4000 Pre-requisites, Support and Restrictions

This section provides the pre-requisites, support information, and restrictions associated with DS4000.

DS4000 Pre-requisites

IBD leverages advanced features of the DS4000. The following Premium Features must be enabled on your DS4000 storage subsystems:

- Host Partitions
- FlashCopy

Two other Premium Features might be required on your DS4000 subsystems, depending on your backup policies and disk configurations:

- VolumeCopy (if VCvolumes are created by your backup policy)
- Intermix (if Fibre Channel and SATA drives are used in your configuration)

DS4000 Support and Restrictions

IBD is supported on DS4000 subsystems with the following requirements and restrictions:

- DS4300 with Storage Manager, V9.1, with Firmware V6.10 or later.
- DS4400 with Storage Manager, V9.1 with Firmware V6.10 or later.
- DS4500 with Storage Manager, V9.1 with Firmware V6.10 or later.
- Up to three DS4000 subsystems can be used in a single IBD association.
- All subsystems in an association must be managed by the same version of the Storage Manager with the same firmware version on each subsystem.
- The subsystems in an association can be different DS4000 models (for example, you can have a DS4300 and a DS4500 in one association).
- Number of LUNs supported on your DS4000 must be at least equal to twice the number of database volumes because database LUNs require corresponding FCvolume LUNs. In addition, two VCvolume LUNs are required for each database that will be backed up using a policy that includes VCvolumes. The DS4300 supports 1024 LUNs, the DS4400 and DS4500 support 2048 LUNs.
- The AIX Failover Driver supports a maximum of 32 LUNs per server.
- For each DB2 instance, an array is required for storing the FCvolumes and a second array is required for storing the VCvolumes if your backup policy for any of the databases in the DB instance creates VCvolumes. You need to set up and allocate sufficient capacity to each of these arrays. If a single database is spread across multiple storage subsystems, arrays are required on each subsystem for FCvolumes and VCvolumes and these arrays (for example, the array for FCvolumes on subsystem 1 for database “A” and the array for the FCvolumes on subsystem 2 for database “A”) must use the same array number.
- The password for a storage subsystem cannot contain a colon, space, or a tab. For example, the password “:xxxyy” or “xx yy” is not allowed.

Note: For information on configuring DS4000 subsystems with a mix of Fibre Channel and SATA disks, refer to the IBM TotalStorage DS4000 Fibre Channel and Serial ATA Intermix Premium Feature Installation Overview at ftp://ftp.software.ibm.com/pc/pccbbs/pc_servers_pdf/gc26771300.pdf.

Chapter 6. Installing IBD

This chapter describes installing, reinstalling and upgrading you Integrated Backup for Databases (IBD) solution.

Pre-installation

Before installing, reinstalling, or upgrading an IBD installation, the following pre-installation steps are required.

1. For each database server in an association, the /etc/hosts file must include the hostname of the database server plus the hostname of the recovery server. The recovery server in each association must include the hostnames of all servers in the association in its /etc/hosts file. For example, if two database servers, with hostnames Joe and Bill, and a recovery server with hostname Mary, are in an association, then the /etc/hosts files must include Joe and Mary on the server named Joe, Bill and Mary on the server named Bill, and Joe, Bill and Mary on the server named Mary. If any hostname is not in the /etc/hosts file, log in as "root" and edit the file to include the hostnames of the database and the recovery servers.
2. For all servers in the association, determine if the rsh daemon is running.
   a. If the rsh daemon is running, stop it using the command:

      ```
      root>stopsrc –t shell
      ```

   b. Append the line “+ + " to the file /etc/hosts.equiv to provide access from any user and from any host, or follow these steps to provide access to selected users on selected hosts.
      For example, assume an association with a database server named Joe that is running a DB2 instance named db2inst1 and a recovery server named Bill.

      In Joe, do the following:
      Append the line “Bill root” in the $HOME/.rhosts file, where $HOME is the home directory of dbinst1. Insert a tab between the words “Bill” and “root.”

      Append the line “Joe db2inst1” in the $HOME/.rhost file, where $HOME is the home directory of root. Insert a tab between the words “Joe” and “db2inst1.”

      In Bill, do the following:

      Append the line “Joe db2inst1” in the $HOME/.rhosts file, where $HOME is the home directory of root.
   c. Log in as "root" and run the following command to restart the rsh daemon on each database server and the recovery server.

      ```
      root>startsrc -t shell
      ```

3. Proceed with the installation as “root” on the recovery server.

New IBD Installation

This section describes installing IBD for the first time in an association. Reinstalling or upgrading IBD is described later in this chapter.

For each IBD association, the solution is first installed on the recovery server and then can be simultaneously installed on all the database servers from the recovery server. An IBD association is a group of one or more database servers and the recovery server that will provide backup services to the associated database servers. A file, named db_bkup_app_2.x.tar, containing the scripts, readme file, and the current version of this user guide can be downloaded from the IBM website.

Recovery Server

Installing IBD on the recovery server is a simple three-step process.

1. Select a directory in the recovery server where you want to install IBD; for example, a directory named IBD under the root directory (/IBD).
2. Go to the /IBD directory using the command:

   root>**cd /IBD**
3. Untar db_bkup_app_2.x.tar (2.x refers to the latest IBD version) using the command on /IBD:

   root>**tar -xvf db_bkup_app_2.x.tar**

This tar command (which untars the file), creates the directory db_bkup_app_2.x under /IBD on the recovery server, which is referred to in this user guide as the Base Directory.

Database Servers

Non-Clustered Database Servers

Installation on the database servers must be performed from the associated recovery server after IBD is installed on the recovery server. From the recovery server, do the following:

1. Change the directory to the Base Directory.

   root>**cd** *Base Directory*
2. Run the install_on_DB_host.sh script specifying the hostname (or the IP address) of the database host and an optional preferred directory as an argument.

   install_on_DB_host.sh –h *dbhostname* [**-p** *pathname*]

Note: *dbhostname* is the hostname of the database server and the optional parameter *pathname* is the absolute path on the database server under which the IBD Base Directory is created. The directory "/db_bkup_app_2.x" (where x refers to the version of IBD) is the default IBD Base Directory on the database server if the *pathname* parameter is not specified.
For example, to install IBD on the path “/backup_dir” in the database server, enter the following command:

root>**install_on_DB_host.sh –h himalaya –p /backup_dir**

High Availability (HA) Clustered Database Servers

In an HACMP cluster, servers are referred to as primary and secondary nodes. The secondary node is the failover node for the primary node. If a primary node fails, its applications fail-over to the secondary node.

When IBD is installed on database servers in an HA cluster, IBD must be installed in the same directory on the primary and secondary servers. Aside from this consideration, installation on the database servers in a cluster is the same for non-clustered servers; follow the preceding process for non-clustered database servers.

IBD is not an HACMP-aware application; however, it will detect a node with IBD-protected databases that fails and will reconnect with the IBD service on the failover node.

Note: The database servers in an association or the database and recovery servers in an association can be configured in a HACMP HA cluster; however, the cluster cannot be configured such that a database server with IBD installed fails-over to the recovery server. IBD-protected DB2 instances cannot run on the recovery server.

Install Script Features

The install_on_DB_host.sh script uses a default path on the database servers. The default path parameter can be changed using the **-p** option.

`install_on_DB_host.sh –h` *`dbhostname`* [**`-p`** *`pathname`*]

IBD can be installed on all the database servers in an association in a single execution by specifying a preferred or default path for each server. For example, if an association contains four database servers, and preferred paths are required for host1 and host2, and default paths for host3 and host4, the following command will install IBD on all four servers:

```
root>install_on_DB_host.sh –h host1 -p path1 -h host2 -p path2
–h host3 -h host4
```

Uninstalling IBD

IBD is uninstalled by running the uninstall_on_DB_host.sh script, which has two options:

–a: This option uninstalls the scripts on all database servers in an IBD association.

```
root>uninstall_on_DB_host.sh –a
```

–h: This option individually uninstalls IBD on a particular database server in an association.

```
root>uninstall_on_DB_host.sh –h host name of the database server
```

For example, to uninstall IBD on database hosts, host1 and host2, enter the following command:

```
root>uninstall_on_DB_host.sh –h host1 –h host2
```

Reinstalling or Upgrading IBD

This procedure applies to situations where it becomes necessary to replace an IBD installation (if for example, an IBD script is accidentally deleted) with the same IBD revision or to upgrade to a new IBD release.

Recovery Server

On the recovery server, do the following:

1. If a backup cycle previously failed, invoke the dbCleanupAll.sh script using the Configuration File run with the failed backup and the resume feature enabled to remove temporary structures, as follows:

   `root>$PWD/`**`dbCleanupAll.sh`** *`configuration file with resume entries`*

2. Run the uninstall_on_DB_host.sh script with the **-a** option specified to uninstall IBD on all database servers associated with this recovery server:

   `root>`**`uninstall_on_DB_host.sh –a`**

   Note: If IBD has to be uninstalled from selected database servers, use the command:

   `uninstall_on_DB_host.sh –h` *`server_name1`* [**`-h`** *`server_ name2`*]

   where server_name1 and *server_name2* are database servers.

   Delete the existing Base Directory and any subdirectories on the recovery server using the command:

   `root>`**`cd /IBD`**

   For example, if the Base Directory is /IBD, the command would be:

   `root>`**`rm -rf db_bkup_app_2.x`**

3. After reviewing the following notes regarding the database servers in the association, go to above Pre-installation section and proceed as if this was a new installation.

   Note: If IBD is installed on a database server and the install_on_DB_host.sh script is run on the recovery server, it will issue a warning that IBD is currently installed on the database server and prompt for confirmation that the existing IBD installation is to be replaced.

   Reinstallation on database servers must be performed from the same recovery server that was used for the previous installation. The install_on_DB_host.sh script maintains installation information on the recovery server. If IBD is reinstalled on a database server from a different recovery server, the install_on_DB_host.sh script will not detect the existing IBD installation and create a second IBD installation on the database server; multiple IBD instances on a server are not allowed.

   If the database servers are configured with HACMP, IBD has to be installed on the failover server in the same directory in which it was previously installed. Follow the steps in the New IBD Installation section for HA clustered database servers in this chapter regarding installation of IBD on a secondary or failover server.

Adding Database Servers to an Association

The install_on_DB_host.sh script is also used to add database servers to an IBD association. The installation process for adding database servers to an association is the same as for the initial set of database servers.

Determine the hostname of the new database servers and perform the following two steps:

1. Change the directory to the Base Directory:

   root>**cd Base Directory**

2. Run the install_on_DB_host.sh script specifying the hostname (or the IP address) of the database host and an optional preferred directory as an argument.

   root>**install_on_DB_host.sh –h** *dbhostname* [**-p** *pathname*]

Note: *dbhostname* is the hostname of the database server and the optional parameter *pathname* is the absolute path on the database server under which the IBD Base Directory is created. The directory "/db_bkup_app_2.x" (where x refers to the version of IBD) is the default IBD Base Directory on the database server if the *pathname* parameter is not specified.
For example, to install IBD on the path “/backup_dir” in the database server, enter the following command:

root>**install_on_DB_host.sh –h himalaya –p /backup_dir**

Replacing the Recovery Server in an Association

Replacing a recovery server in an association requires IBD to be uninstalled on all servers in the association and then performing the same steps as are required for a new installation.

1. Uninstall the IBD on the recovery server. This, in turn, will uninstall IBD on all database servers associated with this server.
2. Identify a new server that can be used as the recovery server.
3. Repeat the pre-installation process (see Pre-Installation section in this chapter).
4. Repeat the installation process (see New IBD Installation section in this chapter).

Chapter 7. Configuring and Running IBD

This chapter describes how to configure the Integrated Backup for Databases (IBD) solution to implement your backup policies and run IBD backup jobs. The Configuration File is the administrative interface to IBD. It can be easily configured to backup one or more databases managed by a DB2 instance and scheduled to run on a user-defined schedule.

Configuration File

If you want to back up all of the databases within a DB2 instance using the same data backup policy and on the same schedule, there is a one-to-one relationship between a Configuration File and a DB2 instance. Figure 15 illustrates this scenario. Each of the three instances can be backed up using a different policy but all of the databases within each instance will be backed up using the same policy. The three instances can be backed up on different schedules but each instance can only have a single backup schedule.

Note: The databases within each instance will be backed up sequentially starting at the scheduled time.

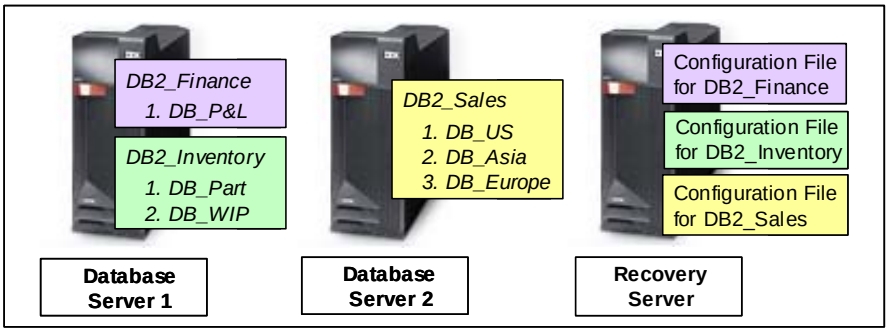


Figure 15. One Configuration File for each DB2 instance

IBD provides the flexibility to backup databases within a DB2 instance using different backup policies as well as to be backed up on different schedules. For either of these cases, multiple configuration files are required for the DB2 instance.

For example, two configuration files (see Figure 16) are required for the DB2_Inventory instance if:

- The backup policy for the DB2_Parts database is to create a VCvolume for quick recovery and an on-tape image for long-term protection, and the backup policy for the DB_WIP database is create just an on-tape image.
  or
- The same backup policy is used for both databases but the schedule requirement is to back up DB_Part daily at 8 AM and DB_WIP daily at 9 PM.

In the first case, where different backup policies are applied to different databases, the Configuration Files are different in two respects: 1) The backup policies are different, and 2) The list of databases to be backed up using each backup policy is different.

In the second case, where different schedule requirements apply to different databases, the Configuration Files are different only in one respect -- the list of databases. In this case, each Configuration File can be attached to separate cron requests to be run at different times.

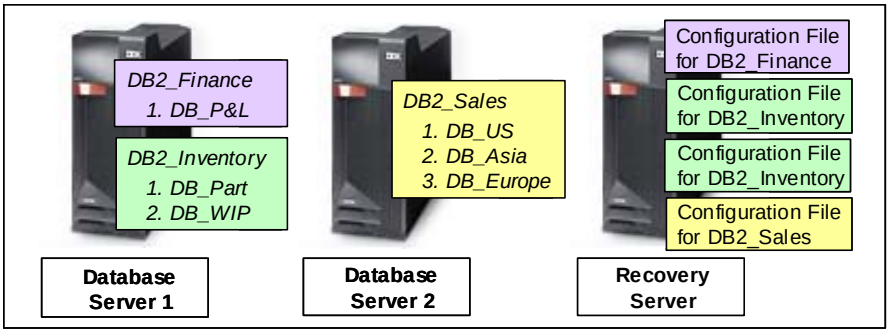


Figure 16. Two configuration files for the DB_Part and DB_WIP

For a single DB2 instance, a Configuration File is required for each combination of:

- Backup policy
- Backup schedule

An example of the number of required Configuration Files for these different combinations of backup policy and backup schedules using the DB2_Inventory instance is shown in Table 4.

Table 4. Example: configuration files needed for the DB2_Inventory instance

Databases	Backup Policy	Backup Cycle	Configuration Files
DB_Part DB_WIP	VolumeCopy and back up to tape	8 PM daily	1 file
DB_Part DB_WIP	VolumeCopy and back up to tape	8 AM daily 9 PM daily	2 files
DB_Part DB_WIP	Back up to tape VolumeCopy and back up to tape	8 AM daily 9 PM daily	2 files

Configuring IBD

Configuring IBD and scheduling it to run automatically is as follows:

- Determine the number of DB2 instances that you want to back up.
- Determine the backup policy for each database within each instance.
- Determine the backup cycle for each database within each instance.
- For each backup policy / backup cycle combination (for each instance), open a Configuration File template.
- For each template, specify values for all mandatory parameters and any optional parameters based on your requirements.

  Note: The template includes default values for some parameters; these can be modified based on your requirements. Table 5 provides a template for Configuration Files.

Table 5. Configuration File template

Parameterstable con (default values in bold)	**Optional / Mandatory**	**Default Value**	**Valid Values**	**Parameter Description**
DB_HOST=	Mandatory		Note 1	Name of HW database server hosting the DB2 instance identified in this Configuration File.
DB_HOST_USERNAME=	Mandatory		Note 1	Username to log into the (above) DB_HOST.
DB_TYPE_NAME=**DB2**	Mandatory	DB2	DB2	DB2 is the only DBMS supported in this release.
DB_INSTANCE=	Mandatory		Note 1	DB2 instance name that owns the databases to be backed up using this Configuration File.
DB_NAME=	Mandatory		Note 6	List of database names to be backed up.
ADDITIONAL_BKUP_FILES_LIST=	Optional		Note 4 or Blank	File (containing the file names with full pathnames) of additional files to backup.
BK_APP_NAME=**TSM**	Mandatory	TSM	Note 5	TSM is the only supported backup/restore application supported in this release.
BK_APP_PASSWORD=	Optional		Note 1 or Blank	Password to access TSM.
BKUP_HOST_PARTITION=	Mandatory		Note 1	DS4000 host group that includes the recovery server as a member. FCvolumes and VCvolumes are mapped to this host partition by IBD.
STORAGE_MANAGER_PASSWORD=	Optional		Note 1 or Blank	A list of passwords for access to each DS4000 Storage Manager in the format “storagesubsystem:password.” A colon must be inserted as the separator between storagesubsystem and password. Spaces are not allowed before or after the colon. If a database resides on multiple storage subsystems, the password for each of the subsystems must be specified in the same line with a tab used as the delimiter. See the example that follows this table. Passwords for a storage subsystem cannot contain a colon, space, or tab. For example, the password, “:xxxyy” or “xx yy” is not allowed.
MOUNT_POINT_HEAD=	See Description		Note 2	Recovery server directory where FCvolumes and VCvolumes are mounted. Mandatory if TSM will be called.
FLASHCOPY_ARRAY_NUM=	Mandatory		0 - 63	DS4000 array # for FCvolumes.
FLASH_REPOSITORY_PERCENT_OF_BASE=**20**	Optional	20	1 - 100	Capacity for FCvolumes expressed as a % of the source volume (integer 20 interpreted as 20%).
ENABLE_VOLUMECOPY=**NO**	Optional	NO	YES, NO, Blank	Specifies if the backup policy includes creating VCvolumes; if NO, IBD automatically calls TSM after creating FCvolumes.
VOLUMECOPY_ARRAY_NUM=	See Description		0 - 63	Mandatory if ENABLE_VOLUMECOPY=YES. DS4000 array number for VCvolumes.
VOLUMECOPY_COPY_PRIORITY=	Optional	Medium	Highest High	Subsystem resource consumption setting for a VolumeCopy operation.

Parameterstable con (default values in bold)	**Optional / Mandatory**	**Default Value**	**Valid Values**	**Parameter Description**
			Medium Low Lowest	"Highest" maximize the performance of the VolumeCopy operation and imposes the highest overhead on the subsystem.
LOG_DIR_PATH=	Optional		Note 3 or Blank	Directory on the recovery server where IBD stores log files.
BACKUP_RETRIES=**3**	Optional		0 – 10 or Blank	TSM retry count (used by IBD if a TSM backup operation fails).
MINUTES_BEFORE_BACKUP _RETRY=**3**	Optional		1 – 10 or Blank	TSM retry wait time (in minutes).
EXIT_ON_NON_OPTIMAL=**NO**	Optional	NO	YES, NO or Blank	Yes = *IBD* will terminate if the subsystem is in a sub-optimal state, for example, a disk rebuild is in progress.
ADMINISTRATOR_MAILID=	Optional		Note 1 or Blank	Full mail address (for notification of critical errors), for example, bill@IBM.com
HACMP_ENABLED=**NO**	Optional	NO	YES, NO or Blank	Indicates whether the database servers are in a high availability (failover) HACMP cluster.
HACMP_RETRY_NUM=	Optional	3	0 – 10 or Blank	Retry count for clustered database nodes. Ignored if HACMP_ENABLE=NO.
HACMP_MIN_BEFORE_RETRY=	Optional	3	1 – 10 or Blank	Node retry wait time (in minutes).

Caution: The IBD Configuration File contains two passwords (BK_APP_PASSWORD= and STORAGE_MANAGER_PASSWORD=) that are used by IBD while it runs. These passwords are not encrypted or otherwise protected. Because this is sensitive information, please ensure that the proper levels of security are set on these files.

STORAGE_MANAGER_PASSWORD= Parameter Example

If a database resides on storagesubsystem1 and storagesubsystem2, the value has to be specified as follows:

```
STORAGE_MANAGER_PASSWORD= subsystem1:password1→subsystem2:password2
```

Where → signifies a tab.

Configuration File Template Notes

The following notes apply to the information in the Valid Values column of Table 5.

Note 1. An alphanumeric string, up to 30 characters in length, containing no spaces or special characters, except for the ADMINISTRATOR_MAILID parameter, which allows the use of special characters. The Configuration File values identified by Note 1 in the Valid Values column are case-sensitive. Values for the following parameters are case-sensitive:

DB_HOST_USERNAME=	STORAGE_MANAGER_PASSWORD=
DB_INSTANCE=	MOUNT_POINT_HEAD=
ADDITIONAL_BKUP_FILES_LIST=	LOG_DIR_PATH=
BK_APP_PASSWORD=	ADMINISTRATOR_MAILID=
BKUP_HOST_PARTITION=	

All of the parameters, such as DB_HOST= are case-sensitive and must be uppercase.

Note 2. IBD will call TSM if:

A. VOLUME_COPY=NO (TSM is called to backup FCvolumes), or

B. If VOLUME_COPY=YES and “-a” run time option specified (TSM is called to backup VCvolumes).

Absolute pathname of the directory or the directory name must be specified; relative pathnames are not allowed.

Note 3 Absolute pathname of the directory or the directory name must be specified; relative pathnames are not allowed. For LOG_DIR_PATH, you can specify a directory other than the Base Directory.

Note 4. Name of a file or the absolute pathname to the file must be specified; relative pathnames are not allowed.

Note 5. This parameter is ignored if ENABLE_VOLUMECOPY is set to YES and the “-a” runtime option is not specified for a regular backup. The “-a” runtime option is used by IBD to determine if TSM should or should not be called after a VolumeCopy operation.

Note 6. The list of database names (DB_NAME=) can be on the same line as the parameter and separated by spaces, or can be listed on multiple lines preceded by DB_NAME=.

The following three parameters have default values that are not shown with the parameter in the Parameters column in Table 5:

- HACMP_RETRY_NUM=
- HACMP_MIN_BEFORE_RETRY=
- VOLUMECOPY_COPY_PRIORITY=

The default values are not shown because IBD ignores these parameters if the parameters they depend on are not enabled:

- HACMP_RETRY_NUM= ignored if HACMP_ENABLE=NO
- HACMP_MINUTES_BEFORE_RETRY= ignored if HACMP_ENABLE=NO
- VOLUMECOPY_COPY_PRIORITY= ignored if ENABLE_VOLUMECOPY=NO

Validating Configuration File Parameters

In addition to validating the values assigned to Configuration File parameters as shown in valid values column of the above template, IBD also validates parameter values for syntactical errors as shown in Table 6.

Table 6. IBD checking for syntactical errors

Parameters	**Description**	**IBD Behavior**
PARAMETER=VAL	*where* VAL is the value assigned to the parameter	Success
PARAMETER =VAL	Tab or insert up to ten spaces before the = character	Success
PARAMETER= VAL	Tab or insert up to ten spaces after the = character	Success
PARAMETER VAL	The = character is deleted	Generates an error
PARAMETER = 0	Value zero (0) is assigned to the variable	Generates an error, except where zero (0) is a valid value. See Valid Values in Table 5.
PARAMETER =	VAL is not specified for an optional parameter	Success
PARAMETER =	VAL is not specified for a mandatory parameter	Generates an error
PARAMETER=VAL space VAL or PARAMETER= VAL PARAMETER= VAL	Two values assigned to a single parameter on the same line, separated by a space	Generates an error, except for the DB_NAME= parameter
PARAMETER= VAL PARAMETER= VAL	Multiple values assigned a single parameter on multiple lines	Generates an error, except for the DB_NAME= parameter

Parameters	Description	IBD Behavior
PARAMETER= VAL tab VAL	Two values assigned to a single parameter on the same line, separated by a tab	Generates an error, except for the STORAGE_MANAGER_PASSWORD= parameter
PARAMETER=INVALID VAL	See Valid Values in Table 5	Generates an error
	Parameter deleted from the Configuration File	Generates an error

Note: Multiple values can be assigned only to the STORAGE_MANAGER_PASSWORD= parameter (in the same line with a tab as the delimiter) and the DB_NAME= parameter (in the same line or in multiple lines with a space as the delimiter).

Configuration File Prerequisites

Before configuring an IBD Configuration File:

- Determine the hostname of each database server that will be included in the IBD solution.
- For each participating DB2 instance, determine the username of the instance and the names of the databases in the instance to be backed up.
- If TSM is called by IBD, determine the TSM password.
- Profile the I/O load on the database server to determine the optimal time periods to run IBD.
- Ensure that your environment conforms to the requirements listed in Chapter 5.

Running IBD

IBD can be run on-demand (invoked from the command line) or on a user-defined backup cycle. To manually invoke IBD for a regular backup, use the following command:

```
root>$PWD/fast_backup.sh –b configuration file [-a] [-d]
```

To run IBD on a regular backup cycle, attach an IBD run request to a cron job list using the **crontab** command; attach the fast_backup.sh script and a Configuration File to the cron job list. Each **cron** command invocation is associated with a specific Configuration File, which is the input parameter for the job. At the scheduled intervals, cron will invoke IBD and begin backing-up the listed databases.

You can attempt to resume an IBD job that previously failed from the command line as:

```
root>$PWD/fast_backup.sh –r configuration file [-d]
```

A backup cycle can fail if:

- An FCvolume has inadequate capacity.
- A VCvolume has inadequate capacity.
- A TSM backup to disk or tape has inadequate capacity.

A resume backup can be attempted to recovery from the last two conditions from the point of failure. This might not always be successful. If for example, a backup cycle fails due to inadequate VCvolume capacity, the FCvolume might also fail before a resumed backup cycle completes.

Note: If a VolumeCopy operation is specified in the backup policy, the FCvolume will be maintained until the VolumeCopy operation completes and while it is being maintained, copy-on-write operations continue to the FCvolume.

Scheduling IBD to Run Automatically

IBD can be scheduled to run on pre-defined schedule by attaching IBD and the Configuration File using the cron Unix scheduler. To attach the IBD job to cron, enter the command:

```
root>crontab -e
```

This command opens the crontab file. The entries for job scheduling are in the format:

minute hour day_of_month month weekday command

The entries can be separated by a space or a tab. After adding the entries, save the file and exit.

If you want the cron to run on multiple days of a month, days must be specified with comma separators, such as 2, 3, 4 for 2nd, 3rd and 4th day of the month. For example, to run the fast_backup.sh script each month on Tuesdays, the weekday column entry is 2. For example:

```
0      22     *     *    2  /rajesh/feb17/db_bkup_app_2.0/fast_backup.sh
–b /rajesh/feb17/db_bkup_app_2.0/config_file -d
```

Note: /rajesh/feb17 in the above command line is the directory where the fast_backup.sh script and the Configuration File for this backup job are stored. The asterisks indicate “all days.”

To list the IBD entries attached to cron, enter the command:

```
root>crontab -l
```

To delete an IBD entry from cron, enter the command:

```
root>crontab –e
```

This command opens the crontab file and lists the IBD entries. You can delete any entry in the file, then save the file and exit.

IBD Command Options

This section describes the IBD command options.

The -a Option (Offloading VCvolumes to Tape)

- If the ENABLE_VOLUMECOPY= parameter is set to NO, IBD calls TSM to back up the FCvolumes.
- If the ENABLE_VOLUMECOPY= parameter is set to YES, then IBD uses the -a option to determine if TSM is called after VCvolumes are created.
  - If the ENABLE_VOLUMECOPY= parameter is set to YES and the -a option is specified for a regular backup, TSM is called to back up the VCvolumes.
  - If the ENABLE_VOLUMECOPY= parameter is set to YES and the -a option is not specified for a regular backup, then TSM is not called and IBD will either start a new database backup cycle (if other databases are queued for backup) or will exit.

When running a resumed backup (see “The –r Option” section) after a regular backup fails, it is not necessary to specify the -a option with the resumed backup. When a backup cycle fails, IBD retains the backup plan that was specified for the failed backup cycle.

If you run a resumed backup and:

- The -a option was specified for the failed backup and the VolumeCopy operation failed, the VolumeCopy operation is resumed and TSM is called.
- The -a option was not specified for the failed backup, the VolumeCopy operation is resumed but TSM is not called.

The -d Option (Debug)

The debug option (-d) enables the collection of additional diagnostic information if *a* backup cycle fails. When the -d option is specified, IBD will not run a cleanup operation that removes the temporary files it creates. These files should be provided to IBM support personnel if assistance resolving a problem is required.

For example, if the –d option is used, the log file contains the message:

"The debug option is set. Therefore, manually delete the temp files in /IBD/db_bkup_app_2.0/.internal/.tempFiles_20050225043418 later."

The temp files can be deleted by entering the command:

```
root>rm –rf /IBD/db_bkup_app_2.0/.internal/.tempFiles_20050225043418
```

The -r Option (Resume)

If IBD is not properly configured, such as allocating sufficient space for FCvolumes or VCvolume, a backup cycle will fail. The resume feature mitigates the work lost due to a failed backup operation by providing the option of resuming the failed backup from the point of failure.

When a backup operation fails, IBD:

- Copies all critical information including the name of the database that was being backed up when the failure occurred to the Restore_Mapfile (metadata).
- Maintains the FlashCopy relationships with the source database volume.
- Retains the VCvolumes (if VolumeCopy was enabled and a TSM operation failed).
- Issues an event notification (email) to the administrator.
- Posts the error to the Event Log, and exit.

If a backup cycle fails, you have three options:

- You can attempt to resume the backup from the point of failure by manually restarting the backup job using the -r (Resume) option.
- You can manually start a new backup using the -b option.
- You can do nothing and back up the database on the next planned cycle.

If you run a resumed backup and the operation succeeds, the database will be backed up as of the same time that it would have been backed up had the original backup not failed. If you manually start a new backup, the backup will be as of the time you invoke the manual backup job. If do nothing, the database will be backed up as of the time of the next scheduled backup cycle. For example, if you discover on Tuesday at 8 AM that the scheduled daily 9 PM backup on Monday failed, you can either:

- Resume the failed backup and attempt to back up the databases as of 9 PM Monday (following the normal scheduled backup cycle).
- Start a regular backup on Tuesday, which will back up the databases as of the time on Tuesday that you invoke the regular backup.
- Ignore the failure and back up the databases at the next scheduled time, (in this case, at 9 PM on Tuesday).

If you resume or initiate a new backup cycle using the same Configuration File, it will remove the entries that were added when it last ran. A failed backup automatically enables the Resume feature. With Resume enabled, the Configuration File cannot be modified until a regular backup (-b option) or a resumed backup (-r option) completes.

Table 7 provides some possible scenarios for the Resume feature.

Table 7. Resume feature scenarios

Failure Case	IBD Behavior
VolumeCopy operation fails	IBD retries the VolumeCopy operation for the failed VCvolume and check the status other VolumeCopy operations for the other FCvolumes. If one or more VCvolumes failed, IBD enables the resume feature. If the -a option was specified for the failed backup, TSM will be called if the VolumeCopy operation completes.
TSM backup operation fails	If VolumeCopy is enabled, IBD will remount the VCvolumes on the recovery server and invoke the TSM to retry the backup operation. If VolumeCopy is not enabled, IBD will remount the FCvolumes on the recovery server and invoke the TSM to retry the backup operation.

Creating a Host Partition for the Recovery Server

A host partition is a logical entity with one or more volumes arranged in arrays that are made accessible to one or more servers. Host partitions allow servers exclusive or shared access to DS4000 volumes. Host-to-volume access is defined by mapping each partition to a single server or a set of servers.

IBD does not require modifications to any existing host-to-volume partition mappings. However, you need to create one additional host partition on each DS4000 subsystem used in an IBD association. For example, if the databases are stored on three DS4000 subsystems, you need to create three host partitions (one on each DS4000) with the same name. You should not map any volumes to these partitions or map these partitions to servers (See Figure 17). IBD will automatically map the FCvolumes and VCvolumes to these partitions and map these partitions to the recovery server, as shown in Figure 18.

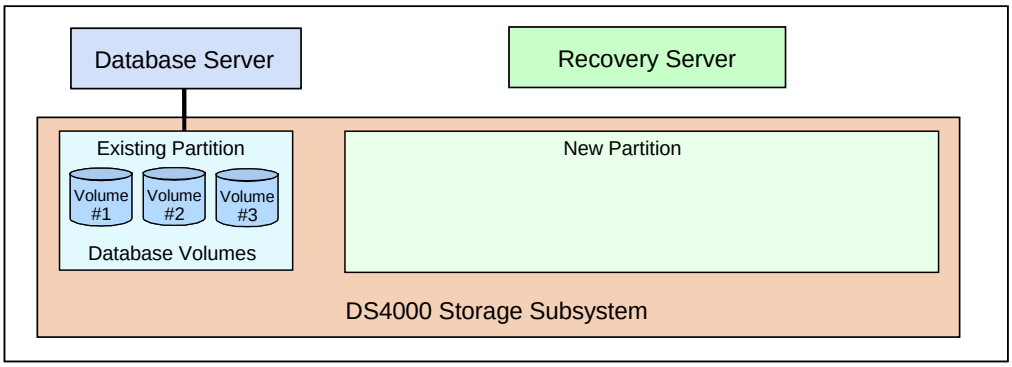


Figure 17.Administrator creates a partition for FCvolumes and VCvolumes

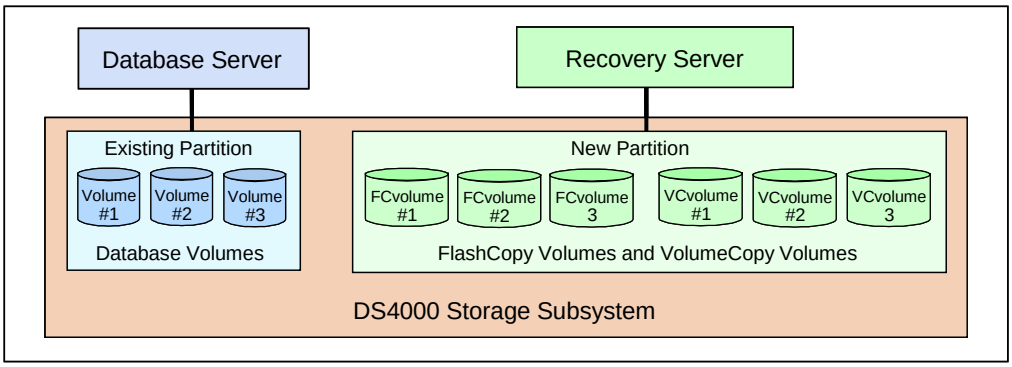


Figure 18. IBD maps the volumes to the partition and the partition to the recovery server.

Recommendations and Restrictions

This section lists IBD configuration recommendations and restrictions.

Recommendations

To enhance performance and IBD operation, follow these guidelines:

- IBD should be schedule to run during periods of relatively low database activity to minimize any impact on application performance.
- An array can contain multiple volumes and should be spread across a large number of high performance physical disks. Generally speaking, database performance increases as more disks are used to store the database.

Restrictions

The following restrictions must be observed:

- If VolumeCopy is enabled, all files to be backed up must be stored on a DS4000.
- Database reconfiguration during an IBD backup cycle is not allowed.
- Modification of the IBD scripts is not allowed and invalidates support.
- The hostname specified in the Configuration File (DB_HOST=) must be the same as the hostname in the /etc/hosts file, which includes the server's IP address. A hostname alias cannot be used in the Configuration File (DB_HOST=).
- The recovery server's hostname and IP address must be specified in the /etc/hosts file on each database server participating in an IBD solution.
- IBD creates volumns in the format, "*database volume*-SNBkup." These volumes should not be mapped to any server.
- If an individual database is spread across multiple subsystems, arrays are required on each subsystem for FCvolumes and VCvolumes, and these arrays must have the same array number on each subsystem.
- If a Configuration File specifies multiple databases stored on multiple subsystems, the partition name used on each subsystem must be the same (this name is specified in the Configuration File using the BKUP_HOST_PARTITION parameter).

Chapter 8. The Database Backup Process

This chapter describes the DB2 split mirror backup strategy and how the Integrated Backup for Databases (IBD) solution uses this DB2 capability to non-disruptively back up databases.

DB2 Split Mirrors

7x24 availability requirements and the size of most databases render traditional tape-based backup methodologies obsolete. Disk-to-disk copy utilities consume substantial server memory bandwidth, as well as server-to-storage fabric bandwidth.

Advanced storage subsystems, such as the DS4000 Series, support Copy Services features that off-load overhead from production servers and SAN fabrics. Because the data moves across back-end storage channels, overhead on production servers and the fabric is eliminated, and the impact of copying data on application latency is reduced.

DB2 uses memory buffer pools to cache tables and index pages as they are read from disk or modified. Bufferpools increase performance because data can read from or written to memory orders of magnitude fasters than from disk. Bufferpools are written to log files and then made permanent in the on-disk database so that partially written transactions can be reapplied or rolled-back. Before an online DB2 database can be backed up using the DB2 split mirror technique, the database must be suspended and brought into a well-defined state.

DB2 implements **set write suspend** commands that IBD invokes to ensure that logical and physical database copies are consistent. By invoking these DB2 commands, IBD forces a database into a well-defined state where a split mirror backup can be initiated. While the database is write-suspended, the database continues to service read I/Os. This allows the split mirror backup process to be transparent to applications and users that remain connected to the database.

IBD Backup Process

IBD backs-up the databases specified in a Configuration File sequentially using the four-phase process illustrated in Figure 19. Backup of each database constitutes an IBD backup cycle.

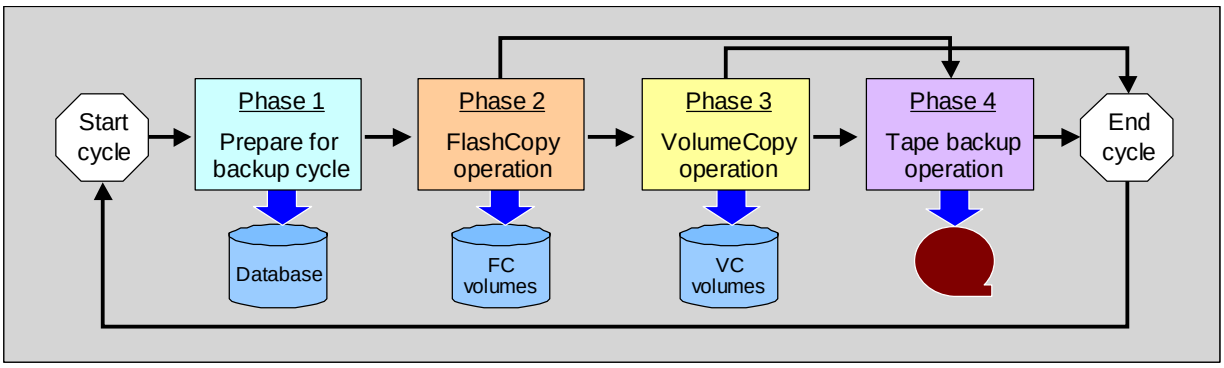


Figure 19. Overview of the backup process

Phase 1: Preparation

During Phase 1, IBD reads and validates a Configuration File, and determines the backup policy to be applied to databases listed in the Configuration File. IBD also determines whether the backup is a regular backup or a resumed backup (if the last backup for this database failed). If this is the first backup cycle for a particular database, IBD creates FCvolumes and VCvolumes if ENABLE_VOLUMECOPY is set to YES. These volumes are then mapped by IBD to recovery server partition.

IBD then creates a backup metadata file for this backup cycle that is called Restore_Mapfile. Pertinent details of the backup process needed for a restore operation will be written to the Restore_Mapfile as the backup progresses through backup phases. If the backup policy is to FlashCopy the database and call TSM, then the Restore_Mapfile is saved with the database in the TSM backup set. For all other backup policies (create VCvolumes or create VCvolumes and back up the VCvolumes to tape), the Restore_Mapfile is appended to the log file that is generated during the backup cycle and is stored on the recovery server.

The final step of Phase 1 is to check the health the DS4000 subsystem. If EXIT_ON_NON_OPTIMAL is set to YES and the subsystem is in a non-optimal state (one of the controllers has failed or a disk rebuild is in progress), IBD will remove temporary files unless the –d (debug) option is specified and exit. If the backup job was run with –d specified, IBD will retain the temporary files and exit.

Phase 2: FlashCopy Operation

IBD invokes the **set write suspend** command for the database to be backed up. IBD then invokes the DS4000's FlashCopy feature to recreate a logical image of the database on FCvolumes. The DS4000 returns control to the operating system immediately after the FlashCopy data structures are set-up. IBD then invokes the DB2 **set write resume** command to allow the resumption of writes to the database. During the time interval between invoking the **set write suspend** and **set write resume** commands, DB2 read I/Os continue to the on-line database and DB2 write I/Os are stored in buffer pools. If the backup policy specified in the Configuration File for the database that is being backed up has ENABLE_VOLUMECOPY set to YES, IBD will move to Phase 3; otherwise, IBD will go directly to Phase 4.

Phase 3: VolumeCopy Operation

In phase 3, IBD invokes the DS4000 VolumeCopy feature and creates a byte-for-byte physical image of the database, called VCvolumes. The VCvolumes can be used as Quick Recovery Volumes if a database restore becomes necessary. Because the VCvolumes are created by copying data from the FCvolumes, they have the same timestamp as the FCvolumes.

VolumeCopy operations can be resource intensive and for this reason, IBD allows you to determine the amount of subsystem resources allocated to this task. The VOLUME_COPY_PRIORITY parameter in the Configuration File determines the amount of subsystem resources allocated to a VolumeCopy operation. The default setting is medium. You can change this setting to high or highest, which will decrease the time required to complete a VolumeCopy operation but impose higher overhead on the subsystem, or you can change this setting to low or lowest, which will increase the time required to complete a VolumeCopy operation but impose less overhead on the subsystem. The FCvolumes are maintained for the duration of the VolumeCopy operation; therefore, you might need to allocate more capacity to the FCvolumes at lower priority VolumeCopy settings. However, depending on the application load imposed on your subsystem, higher VolumeCopy priority setting could impact application performance. The optimal VolumeCopy priority setting is the highest level

that provides acceptable application performance. Some experimentation might be required to determine the optimal setting for your environment.

As mentioned earlier, IBD maintains redundant sets of VCvolumes. If a database corruption occurs during a VolumeCopy operation and is propagated to a VCvolume set that is in the process of being created, the database can be restored from the other VCvolume set.

When the VolumeCopy operation completes, IBD invokes a file system check utility and verifies the consistency of the VCvolumes and then moves to Phase 4 if the backup policy includes a TSM backup.

Phase 4: TSM Operation

In phase 4, IBD calls TSM and provides either the FCvolumes (if ENABLE_VOLUMECOPY is set to NO) or the VCvolumes as input parameters. If TSM returns status indicating that the TSM operation failed, IBD will retry the TSM operation if the BACKUP_RETRIES parameter in the Configuration File is set to a value between one and ten, the number of retries that have already occurred is less than this value.

When Phase 4 completes, IBD will either start the next backup cycle (if there are other databases in the Configuration File that have not been backed up) or exit.

IBD does not define the configuration of a TSM backup. You need to use a TSM interface to configure TSM. IBD simply calls TSM, provides FCvolumes or VCvolumes as input parameters (depending on the backup policy), checks for TSM status, and retries TSM if a TSM operation fails and the TSM retry parameter in the Configuration File is enabled.

The IBD Backup Set

The Backup Set is the collection of files that are backed up by IBD and includes:

- Table spaces
- Database paths
- ADDITIONAL_BKUP_FILES_LIST (an optional parameter in the Configuration File)
- Restore_Mapfile and IBD logs

A *Restore_Mapfile* is a backup metadata file created by IBD for each backup cycle. During a restore operation, this is the first file that is restored and its contents facilitate the restore process, particularly if the database has to be restored to a new hardware server. The Restore_Mapfile is described in Appendix C.

In order for IBD to automatically create a complete backup set, all elements of the backup set must reside on the same DS4000 that stores the databases. All of the database components must reside on the storage subsystem. If any of the additional backup files specified in the Configuration File are stored on a server's internal disks, you must copy (rcp) these files to a specified directory on the recovery server or nfs mount them so the recovery server can see them. These file names along with their full path names must be saved in a file and provided as the value for the ADDITIONAL_BKUP_FILES_LIST parameter in the Configuration File.

We suggest creating databases that are expected to experience rapid growth on logical volumes and, therefore, on filesystem mount points. In addition, if VolumeCopy is enabled, each AIX volume group should contain only one database.

Note: IBD does not backup DB2 archive logs. When IDB invokes the **set write suspend** command, DB2 suspends writes to the database identified as an argument but continues writing to DB2 buffer pools and active logs. If IBD FlashCopied the DB2 archive logs, a race condition (caused by DB2 writing active to archive logs when the FlashCopy operation is invoked) could cause a log file to be partially FlashCopied. If this occurred, it could cause a database corruption when the archive logs are applied to the database during a restore operation. Therefore, you must backup the DB2 logs using TSM if you plan to use roll-forward recovery during a database restore operation.

Chapter 9. The Database Recovery Process

This chapter describes the recovery process. The particularly recovery process that you will use depends on the backup strategy used for the database that needs to be recovered and whether recovery will be to the same database server that managed the database or a different one.

A database can become unusable due to hardware or software faults, or user error. Based on the backup strategy, different recovery strategies are required to deal with various failure scenarios. DB2 supports crash recovery in the event that an online database crashes, causing the database to be in an inconsistent state, but otherwise undamaged. With crash recovery, incomplete transactions are rolled back and committed transactions that were not made permanent to the physical database before the crash are applied to the database from the logs.

For other failure scenarios where a corrupted database must be restored from a previous version of the database, DB2 supports two recovery strategies: Version Recovery, and Roll Forward Recovery.

DB2 Version Recovery

With circular logging enabled, version recovery is the only available DB2 recovery strategy and databases can be restored to a previous version using a backup image. Because archive logging is not enabled, log files (since the last backup) cannot be applied and database transactions since the last backup are unavoidably lost.

DB2 Roll Forward Recovery

With archive logging enabled, roll forward recovery is possible. Databases can be restored to the point of failure without the loss of any transactions since the last backup. A previous version of a database is used to restore the damaged database and logs are applied to the point of failure.

IBD supports version and roll forward recovery strategies, and maintains a Backup Set that is sufficient for recovery to either the database (hardware) server that previously managed the database or to a new (hardware) server in the event that old hardware server is unavailable. Table 8 illustrates the alternative IBD recovery strategies.

Table 8. Restore strategies enabled by IBD

	QRV to Same HW Server	**QRV to a New HW Server**	**TSM Repository to the Same HW Server**	**TSM Repository to a New HW Server**
Version Recovery	Yes	Yes	Yes	Yes
Roll Forward Recovery	Yes	Yes	Yes	Yes

During each database backup cycle, IBD generates a Restore_Mapfile file that contains information (metadata) about the backup and a log file. The Restore_Mapfile file is crucial to the restore process and it is described in Appendix C.

Version Recovery

This section provides the Version Recovery restore procedures; the following four restore scenarios are described:

- From VCvolumes to the same database server
- From VCvolumes to a different database server
- From a TSM Repository to the same database server
- From a TSM Repository to a different database servers.

Version Recovery: QRV to the Same Database Server

Feasibility of Recovery

1. Determine when the database corruption occurred.
2. Identify the most current backup log file (using the **DB_Backup_Log** *Timestamp* command) prior to the corruption by identifying the timestamp. The timestamp is in the format: YYYYMMDDHHMMSS. For example, 20050207141701. The "Restore_Mapfile" section in the backup log file contains the details of VCvolume sets. This VCvolume set will be used to restore and it is referred to in this section as "Restore VCvolume Set."

   Note: Ensure that not more than one backup cycle has taken place since the database corruption occurred. If more than one backup cycle has occurred, then recovery from VCvolumes is not feasible. Refer to the "Version Recovery: TSM Repository to Same Database Server" section in this chapter for information on how to proceed.

Preparation for Recovery

1. Run the enable_restore.sh script on the recovery server with the Restore_Mapfile that corresponds to the Restore VCvolume Set as input. This ensures that the Restore VCvolume Set is not overwritten during a subsequent backup cycle.

   ```
   root>$PWD/enable_restore.sh Restore_Mapfile
   ```

2. Map the VCvolume destination volumes (volumes names are formatted – "VC *timestamp*") from the Restore VCvolume Set to the database server using the DS4000 Storage Manager. The VCvolume names are in the Restore_Mapfile.
3. Run the hot_add utility on the database server:

   ```
   root>hot_add
   ```

4. Run SMdevices on the database server to determine the hdisks corresponding to the VCvolume Recovery Set.

   ```
   root>SMdevices
   ```

For example, when you run SMdevices, hdisks corresponding to each VCvolume are generated. Using this output and the Restore_Mapfile, a table similar to the following table can be created. You can then refer to this table when creating volume groups.

VC Volume Label	Storage Volume Label	VG Name	Hdisks
VC200502082353250	akbar	sultan	hdisk22
VC200502082353251	babur	sultan	hdisk23
VC200502082353252	humayun	sultan	hdisk24
VC200502082353253	aurangazeeb	aurangazeeb	hdisk25
VC200502082353254	shajahan	shajahan	hdisk26
VC200502082353255	jahangir	jahangirVG	hdisk27

Remove the Corrupt Database

1. If clients are using other databases in this instance, disconnect all using the **db2 disconnect all** command.
2. If the instance is running, run the **db2stop force** command as the administrator.
3. If the corrupt database is on a filesystem, unmount the filesystem volumes.
4. Enter the **rmfs** command to remove filesystems and logical volumes from the dismounted volumes.
5. Enter the **varyoffvg** and **exportvg** commands for each volume group that is identified in the Restore_Mapfile.

Restoring the Restore VCvolume Set

1. Run the **recreatevg** command for each volume group that is identified in the Restore_Mapfile by entering the following command, where VG Names are from the Restore_Mapfile and the List of Hdisks are from the output of the **SMdevices** command:

   root>**recreatevg –L** / **-Y NA –y** *VG Name List of Hdisks in the VG*

   For example,

   recreatevg –L / **-Y NA –y** sultan hdisk22 hdisk23 hdisk24
2. If the corrupt database includes raw table spaces, determine if the logical volume names used in the database server (see the Restore_Mapfile file) match the recreated logical volumes. If they do not match, rename the recreated logical volume by entering the command:

   root>**chlv –n** *logical volume name as in the database server of Restore_Mapfile* recreated name
3. Mount the database directory, table spaces and log files in the corresponding directories identified in the Restore_Mapfile.

Restore the Database

1. Ensure that the instance owner has appropriate permissions on the filesystem that contains the database and table space files as well as any device files with raw table spaces.
2. Log in as the DB2 instance owner, enter the **db2start** command, and then initialize the database as a “snapshot” database, as follows:

   db2inidb *database name* **as snapshot**

The initialized databases should now be in a consistent state and available for normal DB2 operations. Connect to the restored database using the **db2 connect to** *database name* command.

Version Recovery: QRV to a Different Database Server

Feasibility of Recovery

1. Determine when the database corruption occurred.
2. Identify the most current backup log file (using the **DB_Backup_Log** *Timestamp* command) prior to the corruption by identifying the timestamp. The timestamp is in the format: YYYYMMDDHHMMSS. For example, 20050207141701. The "Restore_Mapfile" section in the backup log file contains the details of VCvolume sets. This VCvolume set will be used to restore and it is referred to in this section as "Restore VCvolume Set."

   Note: Ensure that not more than one backup cycle has taken place since the database corruption occurred. If more than one backup cycle has occurred, then recovery from VCvolumes is not feasible. Refer to the "Version Recovery: TSM Repository to Same Database Server" section in this chapter for information on how to proceed.

Preparation for Recovery

1. Run the enable_restore.sh script on the recovery server with the Restore_Mapfile that corresponds to the Restore VCvolume Set as input. This ensures that the Restore VCvolume Set is not overwritten during a subsequent backup cycle.

   ```
   root>$PWD/enable_restore.sh Restore_Mapfile
   ```

2. Map the VCvolume destination volumes (volumes names are formatted – "VC *timestamp*") from the Restore VCvolume Set to the database server using the DS4000 Storage Manager. The VCvolume names are in the Restore_Mapfile.
3. Run the hot_add utility on the database server:

   ```
   root>hot_add
   ```

4. Run SMdevices on the database server to determine the hdisks corresponding to the VCvolume Recovery Set.

   ```
   root>SMdevices
   ```

   For example, when you run SMdevices, hdisks corresponding to each VCvolume are generated. Using this output and the Restore_Mapfile, a table similar to the following table can be created. You can then refer to this table when creating volume groups.

VC Volume Label	Storage Volume Label	VG Name	Hdisks
VC200502082353250	akbar	sultan	hdisk22
VC200502082353251	babur	sultan	hdisk23
VC200502082353252	humayun	sultan	hdisk24
VC200502082353253	aurangazeeb	aurangazeeb	hdisk25
VC200502082353254	shajahan	shajahan	hdisk26
VC200502082353255	jahangir	jahangirVG	hdisk27

Restoring the Restore VCvolume Set

1. Run the **recreatevg** command for each volume group that is identified in the Restore_Mapfile by entering the following command, where VG Names are from the Restore_Mapfile and the List of Hdisks are from the output of the **SMdevices** command:

   root>**recreatevg –L** / **-Y NA –y** *VG Name List of Hdisks in the VG*

   For example,

   recreatevg –L / **-Y NA –y** sultan hdisk22 hdisk23 hdisk24

2. If the corrupt database includes raw table spaces, determine if the logical volume names used in the database server (see the Restore_Mapfile file) match the recreated logical volumes. If they do not match, rename the recreated logical volume using the command:

   root>**chlv –n** *logical volume name as in the database server of Restore_Mapfile recreated name*

3. Ensure that the mount point directories in the Restore_Mapfile for databases and table spaces are available in the database server. If not, create the directories using the **mkdir** command.

   root>**mkdir** *mount point dir mentioned in the Restore_Mapfile*

4. Mount the database directory and table spaces in the corresponding directories as identified in the Restore_Mapfile.

Restore the Database

1. Create a DB2 instance with the same Database instance name as is identified in the Restore_Mapfile.
2. Log in as the DB2 instance owner.
3. Ensure that the database instance owner has appropriate permissions on the filesystem that contains the database and table space files.
4. If the database contains raw table spaces, ensure that the database instance owner has appropriate permissions on the device files of the raw table spaces.
5. Catalog the database (system database directory) using the ON parameter. **db2 catalog database** *database-name* **on** *database mount point*
6. Stop the DB2 instance if it is running on the server by using the **db2 terminate** command and then enter the **db2start** command to continue.
7. Initialize the database as a “snapshot” by running the command:

   db2inidb *database name* **as snapshot**

Version recovery to point-in-time of QRV timestamp is now complete and the database should be in a consistent state. Connect to the restored database using the **db2 connect to** *database name* command.

Version Recovery: TSM Storage Pool to the Same Database Server

Preparation for Recovery

- Ensure that the TSM client is configured on both the database and recovery servers such that the databases backed up using the TSM client on the recovery server can be restored using the TSM client running on the database server.

Removing the Corrupt Database

1. If DB2 clients are using other databases in this instance, disconnect all using the **db2 disconnect all** command.
2. If the instance is running, run the **db2stop force** command as the administrator.
3. If the corrupt database is on a filesystem, unmount the filesystem volumes.
4. Use the **rmfs** command to remove the filesystems and logical volumes from the dismounted volumes.
5. Enter the **varyoffvg** and **exportvg** commands for each volume group that is identified in the Restore_Mapfile.

Restore from TSM

1. Restore the Restore_Mapfile (stored with the database backup set in the TSM repository) using the command:

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/Restore_Mapfile destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   dbname refers to the corrupt database.

   Destination directory is the directory where the Restore_Mapfile will be restored on the database server. The Restore_Mapfile contains the database path and table space paths.
2. Obtain the volume group details from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directory for storing the database volumes. Enter the **dsmc restore** command for the database path:

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   LVName is the logical volume name of the database server (in the Restore_Mapfile) where the database path will be restored.

   Destination directory is the mount point where the table space path will be restored to on the database server.

   If the database path was not configured on a mount point, use the directory path name followed by LVNAME in the above command. If the database path is configured under the DB2 instance directory, ensure that you do not overwrite the “sqldbdir” subdirectory under the DB2 instance directory.

3. Obtain the AIX volume group from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directories for storing the table space if they reside on a path other than the database path. Enter the **dsmc restore** command for each table space that resides on a path that is different from the database path.

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   LVName is the logical volume name of the database server (in the Restore_Mapfile) where the table space path was configured.

   Destination directory is the mount point where the table space path will be restored to on the database server.

4. If the backed up table space is raw, run the **dsmc restore image** command to restore the table space:

   dsmc restore image *LVNAME1 LVNAME2*

   where LVNAME1 is the logical volume name for the table space as found in the recovery server in the Restore_Mapfile and LVNAME2: is the logical volume name for the table space as found in the database server without 'r'.

   For example, if "/dev/rmylv" is the logical volume name found in the database server and LV62031604 is the logical volume name found in the recovery server, then the command syntax would be:

   dsmc restore image /dev/LV62031604 /dev/mylv

Restore the Database

1. Log in as the DB2 instance owner and restart the database instance using the **db2start** command.
2. Ensure that the database instance owner has appropriate permissions on the filesystem that contains the database and table space files.
3. If the database contains raw table spaces, ensure that the database instance owner has appropriate permissions on the device files of the raw table spaces.
4. Initialize the database as a "snapshot" by running the command:

   db2inidb *database name* **as snapshot**

Version recovery to point-in-time of the QRV is now complete. The database should be in a consistent state. Connect to the restored database using the **db2 connect to** *database name* command.

Version Recovery: TSM Storage Pool to a Different Database Server

Preparation for Recovery

- Ensure that the TSM client is configured on both the database and recovery servers such that the databases backed up using the TSM client on the recovery server can be restored using the TSM client running on the database server.

Restore from TSM

- Restore the Restore_Mapfile (stored with the database backup set in the TSM repository) by entering the command:

  dsmc restore $MOUNT_POINT_HEAD/$DB_INSTANCE/*dbname*/**Restore_Mapfile** *destination directory/*

Note: Ensure that the destination directory is followed by the slash (/) character.

MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

dbname refers to the corrupt database.

Destination directory is the directory where the Restore_Mapfile will be restored on the database server. The Restore_Mapfile contains the database path and table space paths.

1. Obtain the AIX Volume group from the Restore_Mapfile and create an AIX Volume group and logical volumes. Mount the directory for storing the database volumes. Enter the **dsmc restore** command for the database path:

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   LVName is the logical volume name of the database server (in the Restore_Mapfile) where the database path will be restored.

   Destination directory is the mount point where the table space path will be restored to on the database server.

   If the database path was not configured on a mount point, use the directory path name followed by LVNAME in the above command. If the database path is configured under the DB2 instance directory, ensure that you do not overwrite the “sqldbdir” subdirectory under the DB2 instance directory.

2. Obtain the AIX Volume group from the Restore_Mapfile and create an AIX Volume group and logical volumes. Mount the directories for storing the table space if they reside on a path other than the database path. Enter the **dsmc restore** command for each table space that resides on a path that is different from the database path:

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

LVName is the logical volume name of the database server (in the Restore_Mapfile) where the table space path was configured.

Destination directory is the mount point where the table space path will be restored to on the database server.

3. If the backed up table space is raw, run the **dsmc restore image** command to restore the table space:

   dsmc restore image *LVNAME1 LVNAME2*

   where LVNAME1 is the logical volume name for the table space as found in the recovery server in the Restore_Mapfile and LVNAME2 is the logical volume name for the table space as found in the database server without 'r'.

   For example, if "/dev/rmylv" is the logical volume name found in the database server and LV62031604 is the logical volume name found in the recovery server, then the command syntax would be:

   dsmc restore image /dev/LV62031604 /dev/mylv

Restore the Database

1. Create the same DB2 instance as identified in the Restore_Mapfile.
2. Ensure that the DB2 instance owner has the privileges on the filesystems and the device files.
3. Log in as the DB2 instance owner.
4. Catalog the database (system database directory) with the ON parameter:

   db2 catalog database *database name* **on** *database path*
5. Stop the DB2 instance if it is running on the server by using the **db2 terminate** command and then enter the **db2start** command to continue.
6. Initialize the database as a snapshot by entering the command:

   db2inidb *database name* **as snapshot**
7. The initialized database should be in a consistent state and available for normal DB2 operations. Connect to the restored database using **db2 connect to** *database name* command.

IBD Support for Roll Forward Recovery

This section provides the Roll Forward restore procedures: the following four restore scenarios are described.

- From VCvolumes to the same database server
- From VCvolumes to a different database server
- From a TSM Repository to the same database server
- From a TSM Repository to a different database servers

To avoid a log corruption during the FlashCopy process, IBD does not back up the DB2 archive logs. Therefore, you should ensure that these logs are backed up using TSM and that they are available for roll-forward recovery. In most cases, you will have to retrieve these logs from the backup server and move them to the database server. After you have restored the databases from a VCvolume Set or TSM repository, you can replay the archive logs and restore a consistent database to the point-in-time that the corruption occurred.

Roll Forward Recovery: QRV to Same Database Server

Determine Feasibility of Recovery Strategy

1. Determine when the database corruption occurred.
2. Identify the most current backup log file (using the **DB_Backup_Log** *Timestamp* command) prior to the corruption by identifying the timestamp. The timestamp is in the format: YYYYMMDDHHMMSS. For example, 20050207141701. The "Restore_Mapfile" section in the backup log file contains the details of VCvolume sets. This VCvolume set will be used to restore and it is referred to in this section as "Restore VCvolume Set." IBD maintains only two VCvolume sets.

   Note: Ensure that not more than one backup cycle has taken place since the database corruption occurred. If more than one backup cycle has occurred, then recovery from VCvolumes is not feasible. Refer to the "Version Recovery: TSM Repository to Same Database Server" section in this chapter for information on how to proceed.

Preparation for Recovery

1. Run the enable_restore.sh script on the recovery server with the Restore_Mapfile that corresponds to the Restore VCvolume Set as input. This ensures that the Restore VCvolume set is not overwritten during a subsequent backup cycle:

   ```
   root>$PWD/enable_restore.sh Restore_Mapfile
   ```

2. Map the VCvolume destination volumes (volumes names are formatted – "VC *timestamp*") from the Restore VCvolume Set to the database server using the DS4000 Storage Manager. The VCvolume names are in the Restore_Mapfile.
3. Run the hot_add utility on the database server:

   ```
   root>hot_add
   ```

4. Run SMdevices on the database server to determine the hdisks corresponding to the VCvolume Recovery Set.

   ```
   root>SMdevices
   ```

   For example, when you run SMdevices, hdisks corresponding to each VCvolume are generated. Using this output and the Restore_Mapfile, a table similar to the following table can be created. You can then refer to this table when creating volume groups.

VC Volume Label	Storage Volume Label	VG Name	Hdisks
VC200502082353250	akbar	sultan	hdisk22
VC200502082353251	babur	sultan	hdisk23
VC200502082353252	humayun	sultan	hdisk24
VC200502082353253	aurangazeeb	aurangazeeb	hdisk25
VC200502082353254	shajahan	shajahan	hdisk26
VC200502082353255	jahangir	jahangirVG	hdisk27

Unmount the Corrupt Database

1. If DB2 clients are using other databases in this instance, disconnect all using the **db2 disconnect all** command.
2. If the instance is running, run the **db2stop force** command as the administrator.
3. If the corrupt database is on a filesystem, unmount the filesystem volumes.
4. Enter the **rmfs** command to remove the filesystems and logical volumes from the dismounted volumes.

5. Enter the **varyoffvg** and **exportvg** commands for each volume group that is identified in the Restore_Mapfile.

Restoring the Restore VCvolume Set

1. Run the **recreatevg** command for each volume group that is identified in the Restore_Mapfile by entering the following command, where VG Names are from the Restore_Mapfile and the List of Hdisks are from the output of the **SMdevices** command:

   root>**recreatevg –L** / **-Y NA –y** *VG Name List of Hdisks in the VG*

   For example,

   recreatevg –L / **-Y NA –y** sultan hdisk22 hdisk23 hdisk24

2. If the corrupt database includes raw table spaces, determine if the logical volume names used in the database server (see the Restore_Mapfile file) match the recreated logical volumes. If they do not match, rename the recreated logical volume using the command:

   root>**chlv –n** *logical volume name as in the database server of Restore_Mapfile recreated name*

3. Mount the database directory, table spaces, and log files in the corresponding directories as identified in the Restore_Mapfile.

Restore the Database

1. Ensure that the database instance owner has appropriate permissions on the filesystem that contains the database and table space files.
2. If the database contains raw table spaces, ensure that the database instance has appropriate permissions on the device files of raw table spaces.
3. Log in as the DB2 instance owner on the database server and run the **db2start** command to start the DB2 instance.
4. Initialize the database as a “mirror” database by entering the command:

   db2inidb *database name* **as mirror**

5. Roll forward the archive logs using the command:

   db2 rollforward database *database name* **to end of logs and complete**

6. The initialized database should now be in a consistent state and available for normal DB2 operations. Connect to the restored database using the **db2 connect to** *database name* command.

Roll Forward Recovery: QRV to Different Database Server

Determine Feasibility of Recovery Strategy

1. Determine when the database corruption occurred.
2. Identify the most current backup log file (using the **DB_Backup_Log** *Timestamp* command) prior to the corruption by identifying the timestamp. The timestamp is in the format: YYYYMMDDHHMMSS. For example, 20050207141701. The “Restore_Mapfile” section in the backup log file contains the details of VCvolume sets. This VCvolume set will be used to restore and it is referred to in this section as “Restore VCvolume Set.”

Note: Ensure that not more than one backup cycle has taken place since the database corruption occurred. If more than one backup cycle has occurred, then recovery from VCvolumes is not feasible. Refer to the "Version Recovery: TSM Repository to Same Database Server" section in this chapter for information on how to proceed.

Preparation for Recovery

1. Run the enable_restore.sh script with the Restore_Mapfile that corresponds to the Restore VCvolume Set as input. This ensures that the Restore VCvolume Set will not be overwritten during a subsequent backup cycle.

   root>**$PWD/enable_restore.sh** *Restore_Mapfile*

2. Map the VCvolume destination volumes (volume names are formatted – "VC *timestamp*") from the Restore VCvolume Set to the database server using the DS4000 Storage Manager. The VCvolume names are in the Restore_Mapfile.
3. Run the hot_add utility on the database server:

   root>**hot_add**

4. Run SMdevices on the database server to determine the hdisks corresponding to the VCvolume Recovery Set:

   root>**SMdevices**

   For example, when you run SMdevices, hdisks corresponding to each VCvolume are generated. Using this output and the Restore_Mapfile, a table similar to the following table can be created. You can then refer to this table when creating volume groups.

VC Volume Label	Storage Volume Label	VG Name	Hdisks
VC200502082353250	akbar	sultan	hdisk22
VC200502082353251	babur	sultan	hdisk23
VC200502082353252	humayun	sultan	hdisk24
VC200502082353253	aurangazeeb	aurangazeeb	hdisk25
VC200502082353254	shajahan	shajahan	hdisk26
VC200502082353255	jahangir	jahangirVG	hdisk27

Restoring the Restore VCvolume Set

1. Run the **recreatevg** command for each volume group that is identified in the Restore_Mapfile by entering the following command, where VG Names are from the Restore_Mapfile and the List of Hdisks are from the output of the **SMdevices** command:

   root>**recreatevg –L / -Y NA –y** *VG Name List of Hdisks in the VG*

   For example,

   recreatevg –L / -Y NA –y sultan hdisk22 hdisk23 hdisk24

2. If the corrupt database includes raw table spaces, determine if the logical volume names used in the database server (see Restore_Mapfile) match the recreated logical volumes. If they do not match, rename the recreated logical volume using the command:

   root>**chlv –n** *logical volume name as in the database server of Restore_Mapfile recreated name*

3. Ensure that the mount point directories in the Restore_Mapfile for databases and table spaces are available in the database server. If not, create the directories using the **mkdir** command:

   ```
   root>mkdir mount point dir mentioned in the Restore_Mapfile
   ```

4. Mount the database directory and table spaces to the corresponding mount points that you located or created on the database server.

Restore the Database

1. Create a DB2 instance with the same database instance name as identified in the Restore_Mapfile.
2. Ensure that the database instance owner has appropriate permissions on the filesystem that contains the database and table space files.
3. If the database contains raw table spaces, ensure that the database instance owner has appropriate permissions on the device files of the raw table spaces.
4. Log in as the DB2 instance administrator.
5. Catalog the database (system database directory) with ON parameter:

   db2 catalog database *database-name* **on** *database mount point*

6. Stop the DB2 instance if it is running on the server by using the **db2 terminate** command and then enter the **db2start** command to continue.
7. Initialize the database as a “standby” database by running the **db2initdb** command:

   db2inidb *database name* **as standby**

   Note: This DB2 command initializes the database as a standby and puts the database in a roll-forward pending state.

8. Roll forward the archive logs against the databases:

   db2 rollforward database *database name* **to end of logs and complete**

9. Roll forward recovery to present time is now complete. The database should be in a consistent state. Connect to the restored database using the **db2 connect to** *database name* command.

Roll Forward Recovery: TSM to the Same Database Server

Preparation for Recovery

- Ensure that the TSM client is configured on both the database servers such that the databases backed up using the TSM client on the recovery server can be restored using the TSM client running on the database server.

Remove the Corrupt Database

1. If DB2 clients are using other databases in this instance, disconnect all using the **db2 disconnect all** command.
2. If the instance is running, run the **db2stop force** command as the administrator.
3. If the corrupt database is on a filesystem, unmount the filesystem volumes.
4. Enter the **rmfs** command to remove the filesystems and logical volumes from the dismounted volumes.
5. Enter the **varyoffvg** and **exportvg** commands on each volume group that is identified in the Restore_Mapfile.

Restore from TSM

1. Restore the Restore_Mapfile (stored with the database backup set in the TSM repository) using the command:

   dsmc restore $MOUNT_POINT_HEAD/$DB_INSTANCE/*dbname/ Restore_Mapfile destination directory/*

   Note: Ensure that the destination directory is followed by the slash (/) character.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the configuration file used to back up the database.

   dbname refers to the corrupt database.

   The destination directory is the directory where the Restore_Mapfile will be restored on the database server. The Restore_Mapfile contains the database path and table space paths.

2. Obtain the Volume group details from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directory for storing the database volumes. Enter the **dsmc restore** command for the database path.

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   LVName is the logical volume name of the database server (in the Restore_Mapfile) where the database path will be restored.

   Destination directory is the mount point where the table space path will be restored to on the database server.

   If the database path was not configured on a mount point, use the directory path name followed by LVNAME in the above command. If the database path is configured under the DB2 instance directory, ensure that you do not overwrite the “sqldbdir” subdirectory under the DB2 instance directory.

3. Obtain the volume group details from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directories for storing the table space if they reside on a path other than the database path. Enter the **dsmc restore** command for each table space that resides on a path that is different from the database path.:

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

LVName is the logical volume name of the database server (in the Restore_Mapfile) where the table space path was configured.

Destination directory is the mount point where the table space path will be restored to on the database server.

4. If the backed up table space is raw, run the **dsmc restore image** command to restore the table space:

   dsmc restore image *LVNAME1 LVNAME2*

   where LVNAME1 is the logical volume name for the table space as found in the recovery server in the Restore_Mapfile and LVNAME2 is the logical volume name for the table space as found in the database server without 'r'.

   For example, if "/dev/rmylv" is the logical volume name found in the database server and LV62031604 is the logical volume name found in the recovery server, then the command syntax would be:

   dsmc restore image /dev/LV62031604 /dev/mylv

Restore the Database

1. Ensure the database instance owner has appropriate permissions on the filesystem that contains the database and table space files.
2. If the database contains raw table spaces, ensure the database instance owner has appropriate permissions on the device files of the raw table spaces.
3. Log in as DB2 instance administrator and restart the database instance using the **db2start** command.
4. Initialize the database as a "mirror" database by running the command:

   db2inidb *database name* **as mirror**

   Note: This DB2 command initializes the database as a mirror and puts the database in a roll-forward pending state.
5. Roll forward the logs using the command:

   db2 rollforward database *database name* **to end of logs and complete**

   Note: This DB2 command applies the logs to a database and rolls it forward to the point-in-time when the database became corrupt.

Roll forward recovery to present time is now complete. The database is in a consistent state. Connect to the restored database using the **db2 connect to** *database name* command.

Roll Forward Recovery: TSM to a Different Database Server

Preparation for Recovery

- Ensure that the TSM client is configured on both the database servers such that the databases backed up using the TSM client on the recovery server can be restored using the TSM client running on the database server.

Restore from TSM

1. Restore the Restore_Mapfile (stored with the database backup set in the TSM repository) using the command:

dsmc restore $MOUNT_POINT_HEAD/$DB_INSTANCE/*dbname/ Restore_Mapfile destination directory/*

Note: Ensure that the destination directory is followed by the slash (/) character.

MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

dbname refers to the corrupt database.

Destination directory is the directory where the Restore_Mapfile will be restored on the database server. The Restore_Mapfile contains the database path and table space paths.

2. Obtain the Volume group details from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directory for storing the database volumes. Enter the **dsmc restore** command for the database path.

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

   LVName is the logical volume name of the database server (in the Restore_Mapfile) where the database path will be restored.

   Destination directory is the mount point where the table space path will be restored to on the database server.

   If the database path was not configured on a mount point, use the directory path name followed by LVNAME in the above command. If the database path is configured under the DB2 instance directory, ensure that you do not overwrite the “sqldbdir” subdirectory under the DB2 instance directory.

3. Obtain the volume group details from the Restore_Mapfile and create an AIX volume group and logical volumes. Mount the directories for storing the table space if they reside on a path other than the database path.

   Enter the **dsmc restore** command for each table space that resides on a path that is different from the database path.

   dsmc restore *$MOUNT_POINT_HEAD mentioned in the config file/DB_INSTANCE mentioned in the config file/dbname/LVName/ destination directory/* **-subdir=yes**

   Note: Ensure that the destination directory is followed by the slash (/) character.

   Subdir=yes, which ensures that all subdirectories specified in the *dir path* will be restored.

   MOUNT_POINT_HEAD and DB_INSTANCE are the same as is specified in the Configuration File used to back up the database.

LVName is the logical volume name of the database server (in the Restore_Mapfile) where the table space path was configured.

Destination directory is the mount point where the table space path will be restored to on the database server.

4. If the backed up table space is raw, run the **dsmc restore image** command to restore the table space:
   dsmc restore image *LVNAME1 LVNAME2*

   where LVNAME1 is the logical volume name for the table space as found in the recovery server in the Restore_Mapfile and LVNAME2 is the logical volume name for the table space as found in the database server without 'r'.

   For example, if "/dev/rmylv" is the logical volume name found in the database server and LV62031604 is the logical volume name found in the recovery server, then the command syntax would be:
   dsmc restore image /dev/LV62031604 /dev/mylv

Restore the Database

1. Create the same DB2 instance as identified in the Restore_Mapfile.
2. Ensure that DB2 instance owner has the privileges on the filesystems and the device files.
3. Log in as DB2 instance administrator and start the DB2 instance by running the **db2start** command
4. Catalog the database (system database directory) with the ON parameter:

   db2 catalog database *database name* **on** *database path*
5. Stop the DB2 instance if it is running on the server by using the **db2 terminate** command and then enter the **db2start** command to continue.
6. Initialize the database by entering the command:

   db2inidb *database name* **as standby**
7. Roll forward the database logs:

   db2 rollforward database *database name* **to the end of logs and complete**

   Note: This DB2 command applies the logs to a database and rolls it forward to the point-in-time when the database became corrupt.
8. Roll forward recovery to present time is now complete. The database is in a consistent state. Connect to the restored database using the **db2 connect to** *database name* command.

Chapter 10. High Availability Cluster Support

This chapter describes how the Integrated Backup for Databases (IBD) solution is used in HACMP High Availability (HA) clusters.

HACMP Clusters

HACMP supports:

- High Availability (HA) Clusters (sometimes called mutual takeover clusters)
- Scalable Clusters (sometimes referred to as partitioned clusters)
- High Availability Scalable Clusters (combination of the two)

This release of IBD is supported for use in High Availability Clusters.

HA Clusters

The objective of HA Clusters is to minimize application downtime; nodes automatically takeover from one another when failures occur. The applications on a failed node are automatically restarted on the secondary or failover node. HACMP uses the “share nothing” cluster architecture; every node has physical access to all disk resources but at any point in time, each node has exclusive access to a specific set of resources. When a secondary node takes over a failed node, it also takes ownership of the failed node’s resources.

IBD in HA Clusters

IBD is not an HACMP application but it supports the backup of databases that are configured in an HACMP environment.

IBD runs in a recovery server and in each database server participating in the solution. The recovery server controls overall execution and uses rsh for communications with the database servers. The recovery server identifies database servers by their hostnames, which are translated to IP addresses using the /etc/hosts file.

If a backup cycle is running when the DB2 instance that is being backed up fails, the IBD software on the recovery server will become aware of this situation via failure status and will try to reconnect to the DB2 instance. When the cluster transition completes and the DB2 instance is restarted on the secondary server, IBD on the recovery server can re-establish communications with the database server and complete the backup cycle. The failover process with respect to IBD is summarized as follows:

- IBD is backing up a database that fails.
- Cluster transition (node takeover) occurs.
- The DB2 instance on the failed node is restarted on the secondary node.
- Procedures are implemented to return the databases within the instance to a consistent state (these procedures are not implemented by IBD).
- The DB2 instance is made available for users and applications to reconnect.
- IBD connects to the server now hosting the DB2 instance.
- IBD proceeds with the backup cycle.

The Configuration File has three parameters pertinent to HA clusters:

- HACMP_ENABLED=
- HACMP_RETRY_NUM=
- HACMP_MIN_BEFORE_RETRY=

If you plan to use IBD in a HA cluster, you need to set the HACMP_ENABLE= parameter to YES, and select values for the HACMP_RETRY_NUM= and HACMP_MIN_BEFORE_RETRY= parameters.

The value you assign to HACMP_RETRY_NUM= sets the maximum number of times IBD on the recovery server will attempt to reestablish communications with IBD on the database server. The value assigned to HACMP_MIN_BEFORE_RETRY= sets the wait time between retry attempts.

Chapter 11. Error Codes and Troubleshooting

This chapter provides the Integrated Backup for Databases (IBD) error codes, potential causes of errors, and suggestions for resolving issues (see Table 9).

Table 9. Error codes

Error Code	Descriptions and Possible Causes	Corrective Action
1	An invalid command option was used in the command line. Valid options are **-a**, **-b**, **-d**, and **-r**.	IBD should be invoked only as specified in this document. See Chapter 7 for more information.
66	Internal error indicating a directory command to a specific directory failed.	Retry the failed backup cycle and if it fails a second time, call support.
67	IBD was run by a non-ROOT user.	Log in as root and run IBD.
68	Configuration File is either: • Empty • Nonexistent or • Does not have read permissions • Does not have valid resume entries	Ensure that a valid Configuration File is included in the job request and that it has the appropriate access permission. If a resumed backup fails with error code 68, invoke a regular backup cycle using the **–b** option.
69	An internal error indicating an IBD script has failed.	Check the log file for more information on the cause of the problem. It could be that IBD on the recovery server cannot communicate with IBD on a database server. Take corrective action for problems such as a network issue or a failed server issue. Otherwise, call support.
70	A logical volume manager command has failed.	Ensure that the volume groups are accessible by running any LVM command (such as the **lsvg** *vgname* command.). Ensure that the logical volumes in the volume groups are accessible by running the **lslv** *lvname* command. If you are unable to vary off/export, ensure that the arrays specified in the log file are not varied off and exported.
71	The hot_add (SMutils utility) used to scan for new storage devices has failed on the recovery server.	Ensure that the Fibre Channel HBA driver is properly initialized, and the cables are properly connected and operational. Ensure that the DS4000 SMutils package is installed in the recovery server.
72	An AIX system utility, such as rsh, fsck, or rcp has failed and caused a backup cycle to fail.	Review the log file for more information on the source of the problem and follow the appropriate recovery action: • Ensure that the files have the appropriate permission (refer to the log file). • Ensure that the file system is not full and, if it is full, create sufficient space in the file system for IBD to create the Restore_Mapfile and log files. • Run fsck and ensure that the source volume file system is consistent.

Error Code	Descriptions and Possible Causes	Corrective Action
		• Ensure that the rsh daemon is running and has appropriate permissions (refer to the Pre-installation section in Chapter 6 for more information). • Ensure that hostnames are updated in the /etc/hosts file (refer to Chapter 6 for more information).
73	Internally generated IBD file is missing.	Check log file for more information. Retry the backup cycle and if the problem persists, call support.
74	SMdevices, which runs on the recovery server and is used to map the host devices to corresponding storage volumes has failed.	SMdevices has been verified for compatibility with AIX failover drivers. If you have a failover driver installed on any server in the association, ensure that it is AIX failover driver V5.2.0.50 or later. Determine if the host can see subsystem volumes by entering the **SMdevices** command. Refer to the troubleshooting section in this chapter for more information.
75	Number of BACKUP_RETRIES exceeded the limit specified in the Configuration File.	Ensure that TSM is configured properly and has sufficient disk or tape space for the intended operations.
76	IBD was invoked with missing command options or the options were specified incorrectly. For example, **$PWD/fast_backup.sh** *config_file* **-a** *should be:* **$PWD/fast_backup.sh –b** *config_file* **-a**.	IBD should be invoked as specified in this document (see “IBD Execution Options” in Chapter 7).
77	Unable to create the VCvolume.	Refer to log files for more information. Ensure that sufficient space is available in the storage subsystem for the VCvolume and that the FCvolume has sufficient space. If not, increase the FCvolume capacity threshold and retry the backup cycle.
78	IBD cannot connect to a database.	Refer to IBM DB2 documentation.
79	IBD cannot suspend a database.	Refer to IBM DB2 documentation.
80	IBD is unable to access table space entries for a database.	See IBM DB2 documentation.
82	The create or recreate operation for an FCvolume failed.	Check the log file for more information and then use the DS4000 Storage Manager to ensure that the FCvolumes exist and are in the enabled state. If the problem persists, the FCvolumes can be manually deleted and will be recreated the next time IBD runs.
83	IBD is unable to resume a backup cycle.	This could be caused by a modification to a Configuration File that is enabled for “resume” before a resumed or regular backup cycle with this Configuration File completes. Run dbCleanupAll.sh script to remove all parameters, free resources allocated for resume and run a regular backup.
84	Internal Error. rcp command failed to copy internal files between a database server and the associated recovery server.	Ensure that the complete set of hostnames are available in the /etc/hosts directory and that the files are accessible and have the appropriate

Error Code	Descriptions and Possible Causes	Corrective Action
		access permissions. See the Pre-installation section of Chapter 6. Ensure that the appropriate access permissions are set in the destination directory.
85	IBD received a signal causing termination.	This error could result from pressing control keys while IBD is executing. Rerun the backup cycle without hitting control keys.
87	SMcli call failed.	Check the log file for more information and then connect to the DS4000 using the DS4000 Storage Manager and verify the corresponding volume that caused the SMcli call failed. See the troubleshooting section in this chapter for more information.
88	EXIT_ON_NON_OPTIMAL is set to YES and IBD exited because the DS4000 subsystem.	IBD has run properly. If the problem causing the DS4000 to be non-optimal is caused by a disk rebuild, wait for the rebuild operation to complete. If the problem is caused by a failed subsystem component, call support.
89	Database backup cycle failed.	Refer to the log files for more details on the specific cause of the failure and take appropriate action, such as. Adjusting the VCvolume capacity, TSM configuration, or available space.
90	Recreation of a volume group by the AIX Volume Manager failed.	Ensure that the physical devices (hdisks) are in the available state and that they are not being used in any other volume groups (check the error log). Run **varyoffvg** and **exportvg** to remove old volume group associations with the physical volumes. If necessary, run rmdev to remove the hdisk entry corresponding to the physical volumes.
91	IBD failed because a database volume or a table space is not stored on a DS4000 subsystem.	Databases can be configured on raw devices or file systems but must be stored on a DS4000 subsystem. Locate the database volume or table space and move it to the DS4000.
92	Deleting a previous set of VCvolumes failed.	Check the log file for more information and then connect to the DS4000 using the DS4000 Storage Manager and rerun the SMcli call that failed.
93	Disabling a FCvolume has failed	Check the log file for more information and then connect to the DS4000 using the DS4000 Storage Manager and re-run the SMcli call that failed. See the log file for more details.
100	IBD Configuration File error. DB_HOST= value is invalid.	Mandatory value. Default value – None. Valid values – Hostname of a hardware server running DB2. An alphanumeric string, up to a maximum of 30 characters, containing no spaces or special characters.
101	IBD Configuration File error. DB_HOST_USERNAME= value is invalid.	Mandatory value. Default value – None. Valid values – Username on hardware server running DB2 with system privileges to runs IBD.

Error Code	Descriptions and Possible Causes	Corrective Action
		An alphanumeric string, up to a maximum of 30 characters, containing no spaces or special characters.
102	IBD Configuration File error. BKUP_HOST_PARTITION= value is invalid.	Mandatory parameter. Default value – None. Valid values – An alphanumeric string, up to a maximum of 30 characters, containing no spaces or special characters.
103	IBD Configuration File error. ADMINISTRATOR_MAILID value is invalid.	Optional parameter. Default value – None. Valid values – Mail address (admin@company.com). An alphanumeric string, up to a maximum of 30 characters, containing no spaces.
104	IBD Configuration File error. DB_TYPE_NAME= value is invalid.	Mandatory parameter. Default value – DB2. Valid value – DB2.
105	IBD Configuration File error. BK_APP_NAME= value is invalid.	Mandatory parameter. Default value – TSM. Valid value – TSM.
106	IBD Configuration File error. BK_APP_PASSWORD value is invalid.	Optional parameter. Default value – None. Valid values – An alphanumeric string, up to a maximum of 30 characters, containing no spaces or special characters.
107	IBD Configuration File error. MOUNT_POINT_HEAD= value is invalid.	Mandatory if TSM will be called. Default value – None. Valid values – Absolute pathname of a directory or a directory name.
108	IBD Configuration File error. FLASHCOPY_ARRAY_NUM= value is invalid.	Mandatory parameter. Default value – None. Valid values – 0 to 63.
109	IBD Configuration File error. FLASH_REPOSITORY_PERCENT_OF_BASE= value is invalid.	Optional parameter. Default value – 20. Valid values – 10 to 100.
110	IBD Configuration File error. ENABLE_VOLUMECOPY= value is invalid.	Optional parameter. Default value – No Valid values – Yes, No, True, False, Blank
111	IBD Configuration File error. VOLUMECOPY_ARRAY_NUM= value is invalid.	Mandatory if ENABLE_VOLUMECOPY=Yes Default value – None. Valid values – 0 to 63.
112	IBD Configuration File error. HACMP_ENABLED= value is invalid.	Optional parameter. Default value – No. Valid values – Yes, No, True, False, Blank
113	IBD Configuration File error. HACMP_RETRY_NUM= value is invalid.	Optional parameter. Default value – 3. Valid values – 1 to 10.
114	IBD Configuration File error. HACMP_MIN_BEFORE_RETRY= value is invalid.	Optional parameter. Default value – 3. Valid values – 1 to 10.
115	IBD Configuration File error.	Optional parameter.

Error Code	Descriptions and Possible Causes	Corrective Action
	VOLUMECOPY_PRIORITY= value is invalid.	Default value – Medium. Valid values – Highest, High, Medium, Low, Lowest.
116	IBD Configuration File error. LOG_DIR_PATH= value is invalid.	Optional parameter. Default value – Installation Directory Valid values – Absolute pathname of a directory or a directory name.
117	IBD Configuration File error. BACKUP_RETRIES= value is invalid.	Optional parameter. Default value – 3. Valid values – 1 to 10.
118	IBD Configuration File error. MINUTES_BEFORE_BACKUP_RETRY= value is invalid.	Optional parameter. Default value – 3. Valid values – 1 to 10.
119	IBD Configuration File error. EXIT_ON_NON_OPTIMAL= value is invalid.	Optional value. Default value – Yes. Valid values – Yes, No, True, False, Blank.
120	IBD Configuration File error. ADDITIONAL_BKUP_FILES_LIST= value is invalid.	Optional value. Default value – None. Valid values – Any file name.
121	IBD Configuration File error. DB_INSTANCE= value is invalid.	Mandatory parameter. Default value – None. Valid values – Name of a DB2 instance. An alphanumeric string, up to a maximum of 30 characters, containing no spaces or special characters.
123	IBD Configuration File error. DB_NAME= value is invalid.	Mandatory parameter. Default value – None. Valid values – Database name or list of database names on the same line or separate lines.
124	IBD Configuration File error. STORAGE_MANAGER_PASSWORD= value is invalid.	Optional parameter. Default value – None. Valid values – A list of passwords for access to each DS4000 Storage Manager in the format “storagesubsystem:password.” A colon must be inserted as the separator between storagesubsystem and password. Spaces are not allowed before or after the colon. If the database resides on multiple storage subsystems, the password for each of the storage subsystem must be specified in the same line with tab as the delimiter. For example, if a database resides on storagesubsystem1 and storagesubsystem2, the value has to be specified as follows STORAGE_MANAGER_PASSWORD= subsystem1:password1 subsystem2:password2 The password for a storage subsystem cannot include a period (.). For example, a password such as “wild.horse” is not allowed.

Note: For error codes 100 through 123, see the Configuration File sections in Chapter 7.

Troubleshooting

This section describes two situations we encountered while subjecting IBD to error conditions. You should not encounter these situations, but if you do the following procedures are provided to resolve them.

Error Code #74

An AIX server can host multiple applications and when they do each application is required to clean-up before it exits. Otherwise, it can cause other applications to fail. While testing IBD, we manually created FCvolumes and VCvolumes (independent of IBD) to simulate the activity of another application running on the recovery server. We mapped and unmapped these volumes to the recovery server but did not clean-up after we unmapped them by invoking the **rmdev** command. The result was that in subsequent backup cycles, IBD was unable to see newly added volumes.

Most mappings between logical volumes on storage subsystem and hdisks are generally for long periods of time; they are setup once and remain in place for the life of the volume. But, for FlashCopy and VolumeCopy volumes, the mapping tends to be very short-lived. The normal flow is for the data to be created, for example FlashCopying an existing volume; then that logical volume is mapped to a LUN; and then that LUN is configured on the host. When the host is finished with the data, the configuration for that device is removed and the logical volume unmapped.

The general application flow is:

1. Create the logical volume on the storage subsystem, for example, FlashCopy volume.
2. Map the logical volume to a LUN on the storage subsystem.
3. On the host, run cfgmgr or another utility, such as hot_add (which is only available if the SMclient has been installed on the AIX host)
4. Identify the hdisk entry corresponding to LUN using the **SMdevices** command or the **fget_config –Av** command.
5. Mount a filesystem on the logical volume created from a volume group comprised of one or more hdisks
6. Do application unique initial processing, if any like changing the PVID.
7. Use the data.
8. Stop the program.
9. Dismount any filesystems.
10. Run the **varyoffvg** command on any volume groups.
11. Un-map the logical volume from the LUN on the subsystem.
12. Remove the hdisk entry by running the **rmdev -dl** command.

If step 12 is not performed then the next application that tries to use that same LUN will fail.

Whenever possible, different applications should reserve and use a set of unique LUNs. For example, Application A can use LUNs ranging from 101-110 and Application B can use LUNs in the range 111-120. If the same LUN is being reused, it is possible that an application can fail because another application did not cleanup properly. Use the **fget_config –Av** command to find the hdisk that matches the DS4000 subsystem and LUN. For fget_config, the LUN number is the third field.

For example:

```
User volume group name = 'DS4500_2'
dac0 ACTIVE dac3 ACTIVE
```

```
Disk       DAC LUN Logical Drive
utm 31
hdisk14 dac3 1        Test_1
hdisk15 dac3 2        Appl_1
hdisk18 dac3 3 4-1  Snapshot Volume
hdisk19 dac3 4 5-1 Snapshot Volume
hdisk20  dac90  10
```

In the preceding example, hdisk18 refers to LUN3 of subsystem "DS4500_2".

If the logical volume has been unmapped, but a new logical volume has not yet been mapped to that LUN, then the LUN field will not have a LUN/Logical Drive value at all. So at any point, the **fget_config –Av** command shows either an hdisk that matches the subsystem name and LUN/Logical Drive or an hdisk that has no LUN/Logical Drive value on the subsystem. In the above example, hdisk20 gives the hdisk entry, which is not cleaned up properly when the volume is unmapped.

If IBD fails with the Error Code # 74, it is unable to see an hdisk entry corresponding to the LUN that it has assigned. To solve this problem, you need to perform the following steps manually:

1. Identify the hdiskxx that does not have any Logical drive against it. (For example, hdisk20).
2. Use "rmdev -dl hdiskxx" to attempt to remove the hdisk
3. re-run **cfgmgr** or **cfgmgr -l darXX**, where darXX is the dar with that LUN.
4. If the "rmdev" fails because it reports busy then there is either:
   - An application that is still active with that hdisk assigned.
   - A filesystem that is still mounted on that hdisk.
   - A logical volume that is still varied on using that hdisk.

The steps to complete the cleanup depend on the application that currently has the hdisk assigned but you should know which applications use temporary LUN mapping and how to clean up for those applications.

Error Code #87

We tested the effects of hitting CTL C while IBD was executing. In most cases, IBD handled this situation properly and continued executing. However, we discovered that if TSM was running when CTL C was hit that it caused TSM to fail and TSM retained a SCSI reservation on the volume (either an FCvolume or a VCvolume) that it was copying to tape. If you want to terminate TSM from the command line, the proper way to do so is by hitting the “q” key (not a CTL C).

When we hit CTL C with TSM running, the following entry was made to the log: “Could not start a logical drive copy - The operation cannot complete because the target logical drive entered has a SCSI-2 or persistent reservation placed on it.” If you encounter Error Code #87, the probably cause is a SCSI reservation that has not been released after a failed TSM backup.

To resolve this situation, run the dbCleanupAll.sh script with resume enabled and verify that a proper cleanup was performed. If the dbCleanupAll.sh script is unable to delete the volume (FCvolume or VCvolume), then it has to be deleted manually, as follows:

- Change controller ownership of the logical drive (FCvolume or VCvolume) to the other controller. For example, if it is currently owned by Controller A, change ownership to Controller B).
- Delete the logical drive.
- Rerun the backup cycle.

Chapter 12. Installation and Operational Requirements

Before installing and running the Integrated Backup for Databases (IBD) solution, review the checklist in Table 10 to ensure that all requirements are met.

Table 10. Installation Requirements Checklist

Requirement	Acknowledged
A minimum of two servers must be installed in an IBD association. The supported servers include: eServer p5, models 520, 520 Express, 550 and 570, and pSeries models 615, 630, 650, 655, and 670.	❑
All hardware components, such as Host Bus Adapters, must be on the IBM Hardware Compatibility List.	❑
Each database server must have the following software installed: AIX, V5.2, JFS or JFS 2, AIX Volume Manager V5.2, Maintenance Level 5, DB2, V8.1 or V8.2, TSM Client, V5.2, DS4000 Storage Manager, SMutils and SMcli, and Sendmail.	❑
The recovery server must have the following software installed: AIX, V5.2, JFS or JFS 2, AIX Volume Manager V5.2, Maintenance Level 5, DB2, V8.1 or V8.2, TSM Client and Server, V5.2, DS4000 SMutils and SMcli, and Sendmail.	❑
The AIX Failover Driver V5.2.0.30 is the only supported failover driver and can be installed on any or every server in an association.	❑
Filesystems, and their logs, that are used to store DB2 objects must be exclusively used for storing DB2 objects, and must be stored in the same volume group.	❑
The active and archive logs cannot reside in the same volume group as the corresponding database and its table spaces.	❑
A DS4000 Storage Manager, V9.1 must be installed on a server with access to all of the DS4000 subsystems in an association.	❑
The servers in an IBD association can be configured in an HACMP High Availability (HA) Cluster. HACMP, V4.5 and V5.2, are supported.	❑
Partitioned databases (logical or physical) are not supported.	❑
Direct attach, SAN or HA cluster configurations are supported.	❑
In a cluster configuration, a database server that is being backed up by IBD is not allowed to failover to the recovery server in the association. If the recovery server in a cluster fails, IBD will become unavailable.	❑
FlashCopy and Host Partitions Premium Features must be installed on all DS4000 subsystems in an association.	❑
VolumeCopy Premium Feature must be installed on every DS4000 subsystem that will be used to stored databases that are backed up with a policy that includes creating VCvolumes for quick recovery.	❑
Intermix Premium Feature must be installed on any DS4000 subsystem that includes a mix of Fibre Channel and SATA disks.	❑
Supported subsystems include the DS4300 with Storage Manager V9.1 (FW 6.10 or later), DS4400 with Storage Manager V9.1 (FW 6.10 or later), and DS4500 with Storage Manager V9.1 (FW 6.10 or later).	❑
All subsystems in an association must be at the same Storage Manger revision level and firmware revision level.	❑

Requirement	Acknowledged
Up to three subsystems can be included in an association.	❑
The number of LUNs supported on the DS4000 subsystem must be at least twice the number of database volumes. In addition, two LUNs are required for database that will be backed up using a policy that includes creating VCvolumes. The DS4300 subsystem supports 1024 LUNs, and the DS4400 and DS4500 subsystems support 2048 LUNs.	❑
For each database to be backed up with IBD, allocate sufficient free space for a corresponding FCvolume. The default capacity threshold setting can be adjusted using the FLASH_REPOSITORY_PERCENT_OF_BASE parameter.	❑
Some experimentation might be required to determine the optimal capacity threshold setting for the FCvolumes. Plan a test run for each Configuration File before putting an IBD solution into production.	❑
For each database to be backed up with a policy that includes creating VCvolumes, allocate free space equal to two times the capacity of the database volumes.	❑
For database to be backed up with a policy that includes creating VCvolumes, all files to be backed up must be stored on a DS4000.	❑
Database reconfiguration during an IBD backup cycle is not allowed.	❑
Modification of IBD scripts is not allowed and invalidates support.	❑
The hostname specified in a Configuration File must be the same hostname as the hostname in the /etc/hosts file. A hostname alias cannot be used in the Configuration File.	❑
An array is required on each DS4000 subsystem for FCvolumes and a second array is required on each DS4000 subsystem for VCvolumes if any of the databases it stores will be backed up using a policy that includes creating VCvolumes.	❑
A host partition must be created on each DS4000 subsystem in an association. Volumes should not be mapped to this partition and this partition should not be mapped to any of the servers in the association. IBD automatically maps the FCvolumes and VCvolumes to this partition, and then maps this partition to the recovery server.	❑
If a Configuration File specifies multiple databases (that are stored on multiple DS4000 subsystems) to be backed up, the partition name used on each subsystem must be the same.	❑
If an individual database is spread across multiple subsystems, arrays are required on each subsystem for FCvolumes and VCvolumes, and these arrays must have the same array number.	❑
Databases should be configured on mount points and not directories under it.	❑
If a database contains table spaces that are expected to grow rapidly, the table spaces should be configured on separate mount points instead of the directories under it.	❑
Before installing IBD, you must update the /etc/hosts file on each server in the association. See the Pre-installation section of Chapter 6.	❑
The db_bkup_app_2.x file is first installed on the recovery server and then on the database servers. See Chapter 6 for detailed installation, re-installation, and upgrade instructions.	❑

Requirement	Acknowledged
All mandatory parameters in the Configuration File must have assigned values and optional parameters must also have assigned values based on your environment and backup policies.	❑
To minimize any negative impact on application performance, schedule IBD to run during periods of relatively low database activity.	❑
To improve database performance and/or IBD performance, spread databases and corresponding FCvolumes and VCvolumes across a relatively large number of disk drives.	❑

Appendix A. Configuration Examples

This appendix contains an example that illustrates setting up a Configuration File for a hypothetical environment. While the environment in this example is simplistic, it illustrates the flexibility of the Integrated Backup for Databases (IBD) solution.

Servers

In this example, a customer has three hardware servers; two servers with a DB2 and a recovery server with TSM (see Figure 20).

- Database server (hardware) named Server_Units running DB2
- Database server (hardware) named Server_Dollars running DB2
- Recovery server (hardware) named Server_Save running TSM (client and server)

The DB2 software systems are named:

- DB2_Server_One that has two instances named DB2_Sales and DB2_Returns
- DB2_Server_Two that has two instances named DB2_Revenue and DB2_Profit

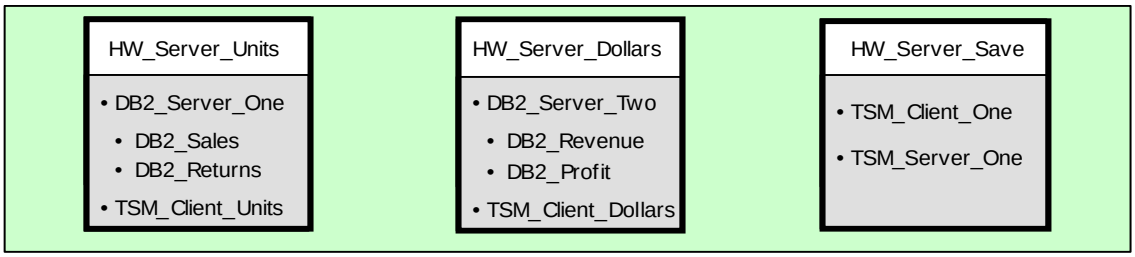


Figure 20. Server environment

Databases

DB2_Sales instance has three databases (see Figure 21), named DB_US, DB_EU and DB_ASIA

DB2_Returns instance has a single database named DB_RMA

DB2_Revenue and DB2_Profit instances have one database each DB_NET and DB_YEAR respectively.

DBMS	Instance	Database
DB_Server_One	DB2_Sales	DB_US
DB_Server_One	DB2_Sales	DB_EU
DB_Server_One	DB2_Sales	DB_ASIA
DB_Server_One	DB2_Returns	DB_RMA
DB_Server_Two	DB2_Revenue	DB_Net
DB_Server_Two	DB2_Profit	DB_Year

Figure 21. Databases

Database Components

Each database has the Table spaces and a database path, as shown in Figure 22.

Database	Table Space	Database Path
DB_US	Vol_TS_US	Vol_PATH_US
DB_EU	Vol_TS_EU	Vol_PATH_EU
DB_ASIA	Vol_TS_ASIA	Vol_PATH_ASIA
DB_RMA	Vol_TS_RMA	Vol_PATH_RMA
DB_Net	Vol_TS_Net	Vol_PATH_Net
DB_Year	Vol_TS_Year	Vol_PATH_Year

Figure 22. Database components

Usernames and Passwords

Passwords for DB2 instances are not shown because they are not required by IBD.

Instances	Usernames	Passwords
DB2_Sales	smith	
DB2_Sales	smith	
DB2_Sales	smith	
DB2_Returns	smith	
DB2_Revenue	gita	
DB2_Profit	gita	
TSM_Client_One	sriram	squire
TSM_Server_One	sriram	squire

Figure 23. Database components

DS4000 Subsystems

In the example shown in Figure 24, the customer has two DS4000 subsystems:

- DS4000 is named Storage_One
- DS4000 is named Storage_Two

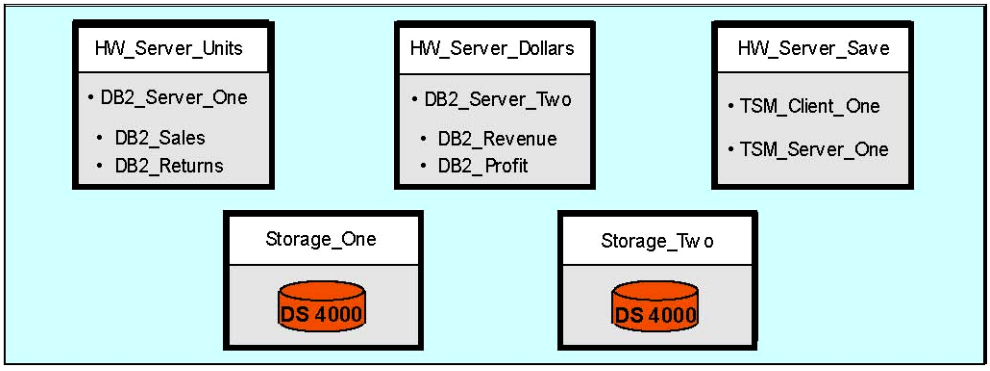


Figure 24. Servers and storage subsystems

Database Mappings

Logical mapping of stored database data managers is shown in Figure 25.

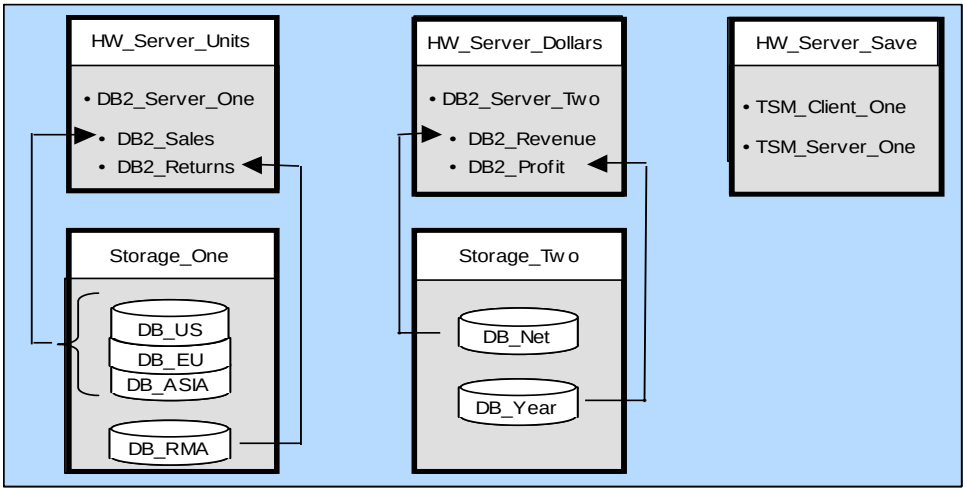


Figure 25. Mappings: Stored databases to data managers

Customer Requirements

Sales and Returns databases are required for business operations. Should these databases become unavailable, the customer is unable to process orders and returns. Therefore, IBD is required to create Quick Recovery Volumes (VCvolumes) and back the data up to tape on a daily basis. Revenue and profit databases are less critical to business operations and can therefore be backed up to tape on a weekly basis (see Figure *26*).

Instance	Database	Requirements	Tape Retry	Exit If Non-optimal
DB2_Sales	DB_US	Create VCvolume, then backup to tape	Yes	No
DB2_Sales	DB_EU	Create VCvolume, then backup to tape	Yes	No
DB2_Sales	DB_ASIA	Create VCvolume only	No	No
DB2_Returns	DB_RMA	Create VCvolume, then backup to tape	No	Yes
DB2_Revenue	DB_Net	Backup to tape	Yes	Yes
DB2_Profit	DB_Year	Backup to tape	Yes	Yes

Figure 26. Customer backup requirements

Configuration File for DB2 Instance DB2_Sales DB Names: DB_US, DB_EU

IBD creates VCvolumes and backups using TSM.

Tape Retry = YES, (IBD would retry 3 times – default.)

MINUTES_BEFORE_RETRY: IBD will retry invoking TSM after every 10 minutes – default

Exit if Non-Optimal=No (IBD would proceed even if storage subsystem is in non-optimal state.)

IBD can set the default path *base directory / backuplogs* as the *LOG_DIR_PATH*. *BASE_DIR* is the directory where the IBD is installed in the recovery server.

The Administrator has created or selected array 7 to be used for storing FlashCopy volumes.

The Administrator has created or selected array 6 to be used for storing VolumeCopy volumes.

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy or VolumeCopy volumes on the recovery server.

Configuration File for Parameters

DB_HOST=DB2_Server_One

DB_HOST_USERNAME=DB2_Sales

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Sales

DB_NAME=DB_US

DB_NAME=DB_EU

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =20

ENABLE_VOLUMECOPY=YES

VOLUMECOPY_ARRAY_NUM =6

VOLUMECOPY_COPY_PRIORITY=medium

LOG_DIR_PATH=

BACKUP_RETRIES=5

MINUTES_BEFORE_BACKUP_RETRIES=

EXIT_ON_NON_OPTIMAL=NO

ADMINISTRATOR_MAILID=sysadmin@ibm.com

HACMP_ENABLED=FALSE

HACMP_RETRY_NUM=

HACMP_MIN_BEFORE_RETRY=

IBD should be invoked from the command line or cron as:

$PWD/fast_backup.sh –b *config_file* **-a**

Configuration File for DB Instance: DB2_Sales DB Name: DB_ASIA

IBD creates VCvolume and uses TSM to offload data to tape.

Tape Retry = YES, BACKUP_RETRIES=5

MINUTES_BEFORE_RETRY=20: (IBD will retry invoking TSM after every 20 minutes for 5 times)

Exit if Non-Optimal=No (IBD will proceed even if storage subsystem is in non-optimal state.)

IBD can set the default path, $BASE_DIR/backuplogs as the LOG_DIR_PATH. BASE_DIR.

This is the directory where IBD is installed in the recovery server.

The Administrator has created array 7 to be used for storing FlashCopy volumes.

The Administrator has created array 6 to be used for storing VolumeCopy volumes.

Configuration File for DB_ASIA

DB_HOST=DB2_Server_one

DB_HOST_USERNAME=DB2_Sales

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Sales

DB_NAME=DB_ASIA

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =20

ENABLE_VOLUMECOPY=YES

VOLUMECOPY_ARRAY_NUM =6

VOLUMECOPY_COPY_PRIORITY=medium

LOG_DIR_PATH=

BACKUP_RETRIES=5

MINUTES_BEFORE_BACKUP_RETRIES=20

EXIT_ON_NON_OPTIMAL=NO

ADMINISTRATOR_MAILID=sysadmin@ibm.com

HACMP_ENABLED=FALSE

HACMP_RETRY_NUM=

HACMP_MIN_BEFORE_RETRY=

IBD should be invoked from the command line or cron as:

$PWD/fast_backup.sh –b *config_file* **-a**

Configuration File for DB Instance DB2_Returns DB Name: DB_RMA

IBD creates VCvolume and backs-up using TSM:

Tape Retry = No (IBD will invoke TSM only once. No retries if backup TSM fails.)

Exit if Non-Optimal=YES (IBD will not proceed if storage subsystem is in non-optimal state.)

IBD can set the default path, *base directory* / *backuplogs* as the *LOG_DIR_PATH*. BASE_DIR is the directory where the IBD is installed on the recovery server.

The Administrator has created array 7 to be used for storing FlashCopy volumes.

The Administrator has created array 6 to be used for storing VolumeCopy volumes.

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy or VolumeCopy volumes on recovery server.

Configuration File Parameters for Database DB_RMA

DB_HOST=DB2_Server_one

DB_HOST_USERNAME=DB2_Returns

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Returns

DB_NAME=DB_RMA

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =30

ENABLE_VOLUMECOPY=YES

VOLUMECOPY_ARRAY_NUM =6

VOLUMECOPY_COPY_PRIORITY=medium

LOG_DIR_PATH=

BACKUP_RETRIES=0

MINUTES_BEFORE_BACKUP_RETRIES=0

EXIT_ON_NON_OPTIMAL=YES

ADMINISTRATOR_MAILID=sysadmin@ibm.com

HACMP_ENABLED=FALSE

HACMP_RETRY_NUM=

HACMP_MIN_BEFORE_RETRY=

IBD should be invoked from the command line or cron as:

$PWD/fast_backup.sh –b *config_file* **-a**

Configuration File for DB Instance DB2_Revenue DB Name: DB_Net

IBD does not create VCvolumes but backs-up using TSM:

Tape Retry = YES, BACKUP_RETRIES=5

MINUTES_BEFORE_RETRY=20: IBD will retry invoking TSM every 20 minutes.

Exit if Non-Optimal=YES (IBD will not proceed if storage subsystem is in non-optimal state.)

IBD can set the default path, $BASE_DIR/backuplogs as the LOG_DIR_PATH. BASE_DIR is the directory where IBD is installed in the recovery server.

IBD does not need to backup Archive Logs for the databases.

The Administrator has created array 7 to be used for storing flash copy volumes.

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy volumes on recovery server.

The Administrator provides the TSM password “squire” in the Configuration File.

DB_HOST=DB2_Server_Two

DB_HOST_USERNAME=DB2_Revenue

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Revenue

DB_NAME=DB_Net

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=squire

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =30

ENABLE_VOLUMECOPY=NO

VOLUMECOPY_ARRAY_NUM =

VOLUMECOPY_COPY_PRIORITY=

LOG_DIR_PATH=

BACKUP_RETRIES=5

MINUTES_BEFORE_BACKUP_RETRIES=20

EXIT_ON_NON_OPTIMAL=YES

ADMINISTRATOR_MAILID=sysadmin@ibm.com

HACMP_ENABLED=FALSE

HACMP_RETRY_NUM=

HACMP_MIN_BEFORE_RETRY=

IBD should be invoked from the command line or cron as:

$PWD/fast_backup.sh –b *config_file*

Configuration File for DB Instance DB2_Profit DB Name: DB_Year

Tape Retry = YES, BACKUP_RETRIES=5

MINUTES_BEFORE_RETRY=20: IBD will retry invoking TSM every 20 minutes.

Exit if Non-Optimal=YES (IBD will not proceed if storage subsystem is in non-optimal state.)

IBD can set the default path, $BASE_DIR/backuplogs as the LOG_DIR_PATH. BASE_DIR is does not need to the directory where the IBD is installed on the recovery server.

The Administrator has created array 7 to be used for storing flash copy volumes

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy volumes on recovery server

The Administrator provides the TSM password “squire” in the Configuration File.

Configuration Parameters for Database DB_Year

DB_HOST=DB2_Server_Two

DB_HOST_USERNAME=DB2_Profit

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Profit

DB_NAME=DB_Year

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=squire

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =30

ENABLE_VOLUMECOPY=NO

VOLUMECOPY_ARRAY_NUM =

VOLUMECOPY_COPY_PRIORITY=

LOG_DIR_PATH=

BACKUP_RETRIES=5
MINUTES_BEFORE_BACKUP_RETRIES=20
EXIT_ON_NON_OPTIMAL=YES
ADMINISTRATOR_MAILID=sysadmin@ibm.com
HACMP_ENABLED=FALSE
HACMP_RETRY_NUM=
HACMP_MIN_BEFORE_RETRY=
IBD should be invoked from the command line or cron as:
$PWD/fast_backup.sh –b *config_file*

Appendix B. HA Cluster Example

This example illustrates a Configuration File for an environment with a 2-node HA cluster.

Servers

There are three hardware servers; the two DB2 servers are clustered.

- Database server (HW) named Server_Units running DB2
- Database server (HW) named Server_Dollars running DB2
- Recovery server (HW) named Server_Save running TSM (client and server)

DB2 systems (software systems) are named as follows:

- There are two instances of DB2 named DB2_Sales_One and DB2_Sales_Two

Figure 27. Server environment

Databases

- DB2_Sales_One instance has 2 databases, DB_US and DB_EU on DB2_Server_One.
- DB2_Sales_Two instance has 1 database, DB_ASIA on DB2_Server_Two.

The servers are configured so that DB2_Server_One will take over for DB2_Server_Two if it fails and vice versa:

- If DB2_Server_One fails, ownership of databases DB_US and DB_EU transfer to DB2_Server_Two.
- If DB2_Server_Two fails, ownership of database DB_ASIA transfers to DB2_Server_One.

DBMS	Instance	Database
DB_Server_One	DB2_Sales_One	DB_US
DB_Server_One	DB2_Sales_One	DB_EU
DB_Server_Two	DB2_Sales_Two	DB_ASIA

Figure 28. Databases

Database Components

Each database has a table space and a database path, as shown in Figure 29.

Database	Table Space	Database Path
DB_US	Vol_TS_US	Vol_PATH_US
DB_EU	Vol_TS_EU	Vol_PATH_EU
DB_ASIA	Vol_TS_ASIA	Vol_PATH_ASIA

Figure 29. Database components

DS4000 Subsystems

The customer has two DS4000 subsystems:

- DS4000 is named Storage_One
- DS4000 is named Storage_Two

Database Mappings

Mapping of stored databases to database mangers is shown in Figure 30.

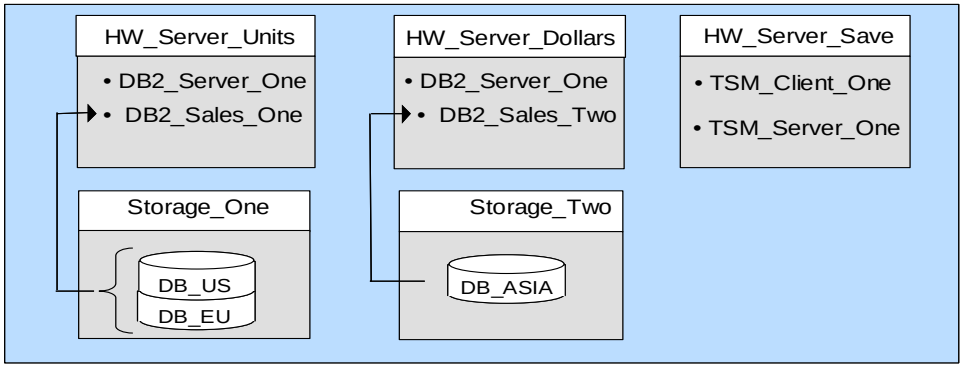


Figure 30. Mapping of stored database to database managers

Customer Requirements

Sales and Returns databases are required for business operations; if these databases become unavailable, the customer cannot process orders and returns. Therefore, the customer requires that IBD create Quick Recovery Volumes (VCvolumes) and also back the data up to tape. The revenue and profit databases are less critical and only need to be backed up to tape weekly.

DBMS	Database	Requirements	Tape Retry	Exit Non-optimal
DB_Server_One	DB_US	Create VCvolume, backup to tape	Yes	No
DB_Server_One	DB_EU	Create VCvolume, backup to tape	Yes	No
DB_Server_Two	DB_ASIA	Create VCvolume	No	No

Figure 31. Customer backup requirements

Configuration Parameters for DB_US and DB_EU

Tape Retry = YES, (IBD will retry 3 times – default.).

MINUTES_BEFORE_RETRY: IBD will retry invoking TSM after every 10 minutes (default).

Exit if Non-Optimal=No (IBD will proceed if storage subsystem is in non-optimal state).

IBD can set the default path, *base directory* / *backuplogs* as the *LOG_DIR_PATH*. BASE_DIR is the directory where the IBD is installed in the Recovery Server.

The Administrator has created array 7 to be used for storing FlashCopy volumes.

The Administrator has created array 6 to be used for storing VolumeCopy volumes.

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy or VolumeCopy volumes on recovery server.

The databases are in HACMP environment. If the database server fails, IBD will retry the operation 5 times every 5 minutes.

Configuration File for DB_US and DB_EU:

DB_HOST=DB2_Server_One

DB_HOST_USERNAME=DB2_Sales_One

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Sales_One

DB_NAME=DB_US

DB_NAME=DB_EU

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =20

ENABLE_VOLUMECOPY=YES

VOLUMECOPY_ARRAY_NUM = 6

VOLUMECOPY_COPY_PRIORITY= medium

LOG_DIR_PATH=

BACKUP_RETRIES=

MINUTES_BEFORE_BACKUP_RETRIES=

EXIT_ON_NON_OPTIMAL=NO ADMINISTRATOR_MAILID=sysadmin@ibm.com

HACMP_ENABLED=TRUE

HACMP_RETRY_NUM=5

HACMP_MIN_BEFORE_RETRY=5

IBD should be invoked from the command line or cron as:

$PWD/fast_backup.sh –b *config_file* **-a**

Configuration Description for the Database, DB_ASIA

Tape Retry = YES, BACKUP_RETRIES=5

MINUTES_BEFORE_RETRY=20: (IBD will retry invoking TSM every 20 minutes 5 times).

Exit if Non-Optimal=No (IBD will proceed if storage subsystem is in non-optimal state).

IBD can set the default path, $BASE_DIR/backuplogs as the LOG_DIR_PATH. BASE_DIR is the directory where the IBD is installed on the recovery server.

The Administrator has created array 7 to be used for storing FlashCopy volumes.

The Administrator has created array 6 to be used for storing VolumeCopy volumes.

The Administrator has created a new mount point, /mnt to be used by IBD for mounting FlashCopy or volume copy volumes on recovery server.

HACMP is enabled. If DB2_Server_Two fails, Database DB_ASIA will be owned by DB2_Server_One. IBD will retry after 3 minutes for 3 times.

Configuration File for DB_ASIA

DB_HOST=DB2_Server_Two

DB_HOST_USERNAME=DB2_Sales_Two

DB_TYPE_NAME=DB2

DB_INSTANCE=DB2_Sales_Two

DB_NAME=DB_ASIA

ADDITIONAL_BKUP_FILES_LIST=

BK_APP_NAME=TSM

BK_APP_PASSWORD=

BKUP_HOST_PARTITION=Server_Save

STORAGE_MANAGER_PASSWORD=

MOUNT_POINT_HEAD=/mnt

FLASHCOPY_ARRAY_NUM =7

FLASH_REPOSITORY_PERCENT_OF_BASE =20

ENABLE_VOLUMECOPY=YES

VOLUMECOPY_ARRAY_NUM =6

VOLUMECOPY_COPY_PRIORITY=medium

LOG_DIR_PATH=
BACKUP_RETRIES=5
MINUTES_BEFORE_BACKUP_RETRIES=20
EXIT_ON_NON_OPTIMAL=NO ADMINISTRATOR_MAILID=sysadmin@ibm.com
HACMP_ENABLED=TRUE
HACMP_RETRY_NUM=
HACMP_MIN_BEFORE_RETRY=
IBD should be invoked from the command line or cron as:
$PWD/fast_backup.sh –b *config_file*

Appendix C. Sample Restore_Mapfile and IBD Log File

Sample Restore_Mapfile

The Restore_Mapfile provides the details to the user for restoring the database. It provides details on how the database and its table spaces are configured on the volumes of the storage subsystems. More details are embedded in the following Restore_Map file sample.

```
Restore Mapfile Contents
Backup performed on Feb-25-2005 at 04:34:56 AM.
********
The below lines indicate that the database, flowers belonging to
db instance, db21inst is created on mount point, /Camelia.
/Camelia is laid on a logical volume Camelialv of the
volumegroup, Cameliavg. Cameliavg is made of 3 storage volumes,
Camelia, Flora and Hana of Storage Subsystem, Indi57.
********
Database: flowers in /Camelia/db21inst/NODE0000/SQL00001/
Database instance name: db21inst
Storage Volume label: Camelia
Storage Subsystem name: Indi57
Storage Volume label: Flora
Storage Subsystem name: Indi57
Storage Volume label: Hana
Storage Subsystem name: Indi57
Database Path: /Camelia   Camelialv   Cameliavg    LV025043418

********
The lines below indicate that the tablespace 0 is on the same
path as that of the database path.
********
TABLE SPACE DETAILS:

Dir path for table space 0 is
/Camelia/db21inst/NODE0000/SQL00001/SQLT0000.0
Tablespace ID 0: /Camelia   Camelialv   Cameliavg    LV025043418

TABLE SPACE DETAILS:

Dir path for table space 1 is
/Camelia/db21inst/NODE0000/SQL00001/SQLT0001.0
Tablespace ID 1: /Camelia   Camelialv   Cameliavg    LV025043418
```

```
TABLE SPACE DETAILS:

Dir path for table space 2 is
/Camelia/db21inst/NODE0000/SQL00001/SQLT0002.0
Tablespace ID 2: /Camelia   Camelialv   Cameliavg    LV025043418

TABLE SPACE DETAILS:

Dir path for table space 3 is /Camelia/t1
Tablespace ID 3: /Camelia   Camelialv   Cameliavg    LV025043418

TABLE SPACE DETAILS:

Dir path for table space 4 is /Camelia/t2
Tablespace ID 4: /Camelia   Camelialv   Cameliavg    LV025043418
********
the lines below indicate that the tablespace 5 and tablespace 6
are created on the mountpoint /Michie under the directories t3
and t4. /Michie is laid on logical volume fslv01 of volumegroup,
Michievg. Michie is made of the physical volume, Michie on the
storage subsystem, Indi57.
********
TABLE SPACE DETAILS:

Dir path for table space 5 is /Michie/t3
Database instance name: db21inst
Storage Volume label: Michie
Storage Subsystem name: Indi57
Tablespace ID 5: /Michie   fslv01   Michievg    LV325043418

TABLE SPACE DETAILS:

Dir path for table space 6 is /Michie/t4
Tablespace ID 6: /Michie   fslv01   Michievg    LV325043418

TABLE SPACE DETAILS:

Dir path for table space 7 is /Jasmine/t5
Database instance name: db21inst
Storage Volume label: Jasmine
Storage Subsystem name: Arjuna
Tablespace ID 7: /Jasmine   jasminelv1   jasminevg   LV425043418
```

```
TABLE SPACE DETAILS:

Dir path for table space 8 is /Jasmine/t6
Tablespace ID 8: /Jasmine   jasminelv1   jasminevg   LV425043418

TABLE SPACE DETAILS:

Dir path for table space 9 is /Jasmine1/t7
Tablespace ID 9: /Jasmine1   jasminelv2   jasminevg  LV525043418

TABLE SPACE DETAILS:

Dir path for table space 10 is /Jasmine1/t8
Tablespace ID 10: /Jasmine1  jasminelv2  jasminevg   LV525043418

*********
Below lines indicate that the tablespace 11 is created on a raw
interface (/dev/rLilylv) of the logical volume, Lilylv.  Lilylv
belongs to the volumegroup, Lilyvg which is made of the storage
volume, Lily of the storage subsystem, Arjuna.
*********
TABLE SPACE DETAILS:

Dir path for table space 11 is /dev/rLilylv
Database instance name: db21inst
Storage Volume label: Lily
Storage Subsystem name: Arjuna
Tablespace ID 11: /dev/rLilylv   Lilylv   Lilyvg    LV725043418

********
The below lines give the details of each and every AIX volume
group, which would be required for restore from TSM backup to a
different server.
********


VG details for Cameliavg
Cameliavg:
LV NAME    TYPE    LPs    PPs    PVs   LV STATE     MOUNT POINT
Camelialv  jfs     1000   1000   2     open/syncd   /Camelia
loglv13    jfslog  1      1      1     open/syncd   N/A
```

```
VG details for Michievg
Michievg:
LV NAME    TYPE       LPs PPs    PVs   LV STATE     MOUNT POINT
loglv14    jfs2log 1      1      1     open/syncd   N/A
fslv01     jfs2    512    512    1     open/syncd   /Michie

VG details for jasminevg
jasminevg:
LV NAME    TYPE    LPs    PPs    PVs   LV STATE     MOUNT POINT
jasminelv1 jfs     300    300    1     open/syncd   /Jasmine
jasminelv2 jfs     300    300    1     open/syncd   /Jasmine1
loglv15    jfslog  1      1      1     open/syncd   N/A

VG details for Lilyvg
Lilyvg:
LV NAME    TYPE    LPs    PPs    PVs   LV STATE     MOUNT POINT
Lilylv     jfs     630    630    1     closed/syncd N/A

***********
Below lines give the mapping of storage volume and the
corresponding volumecopy volume.
***********
Mapping Information for VolumeCopy volumes:

Storage Volume Label: Camelia
VolumeCopy Volume Label: VC200502250434180
Storage subsystem: Indi57

Storage Volume Label: Flora
VolumeCopy Volume Label: VC200502250434181
Storage subsystem: Indi57

Storage Volume Label: Hana
VolumeCopy Volume Label: VC200502250434182
Storage subsystem: Indi57

Storage Volume Label: Michie
VolumeCopy Volume Label: VC200502250434183
Storage subsystem: Indi57

Storage Volume Label: Jasmine
VolumeCopy Volume Label: VC200502250434184
Storage subsystem: Arjuna
```

```
Storage Volume Label: Lily
VolumeCopy Volume Label: VC200502250434185
Storage subsystem: Arjuna


    *** END OF RESTORE MAP FILE ***
```

Sample Log File

Integrated Backup for Databases

Database Backup Log created on Feb-25-2005 at 04:34:18 AM.

WARNING:config_file8 does not have a valid entry for ADMINISTRATOR EMAIL ID or there is more than one entry in the given configfile

ADMINISTRATOR EMAIL ID. The Administrator cannot be notified

in case of critical errors.

Attempting to backup the database "flowers"...

Setting up the configuration required for

rsh done successfully

Connecting to Database. flowers

Connected successfully to database. flowers

Executing the SMdevices command on localhost.

Finding the mount point for Database directory.

*** /Camelia Camelialv Cameliavg ***

Finished processing the Database directory

Finding the mount point for Tablespace ID 0.

Getting the path for Tablespace ID 0

verifying the path with the mount command for Tablespace ID 0

and the container ID to 0.

*** /Camelia Camelialv Cameliavg ***

Written to the map_file for Tablespace ID 0.

Finished processing the Tablespace ID 0.

Finding the mount point for Tablespace ID 1.

Getting the path for Tablespace ID 1

verifying the path with the mount command for Tablespace ID 1

and the container ID to 0.

*** /Camelia Camelialv Cameliavg ***

Written to the map_file for Tablespace ID 1.

Finished processing the Tablespace ID 1.

Finding the mount point for Tablespace ID 2.

Getting the path for Tablespace ID 2

verifying the path with the mount command for Tablespace ID 2
and the container ID to 0.
*** /Camelia Camelialv Cameliavg ***
Written to the map_file for Tablespace ID 2.
Finished processing the Tablespace ID 2.
Finding the mount point for Tablespace ID 3.
Getting the path for Tablespace ID 3
verifying the path with the mount command for Tablespace ID 3
and the container ID to 0.
*** /Camelia Camelialv Cameliavg ***
Written to the map_file for Tablespace ID 3.
Finished processing the Tablespace ID 3.
Finding the mount point for Tablespace ID 4.
Getting the path for Tablespace ID 4
verifying the path with the mount command for Tablespace ID 4
and the container ID to 0.
*** /Camelia Camelialv Cameliavg ***
Written to the map_file for Tablespace ID 4.
Finished processing the Tablespace ID 4.
Finding the mount point for Tablespace ID 5.
Getting the path for Tablespace ID 5
verifying the path with the mount command for Tablespace ID 5
and the container ID to 0.
*** /Michie fslv01 Michievg ***
Written to the map_file for Tablespace ID 5.
Finished processing the Tablespace ID 5.
Finding the mount point for Tablespace ID 6.
Getting the path for Tablespace ID 6
verifying the path with the mount command for Tablespace ID 6
and the container ID to 0.
*** /Michie fslv01 Michievg ***
Written to the map_file for Tablespace ID 6.
Finished processing the Tablespace ID 6.
Finding the mount point for Tablespace ID 7.
Getting the path for Tablespace ID 7
verifying the path with the mount command for Tablespace ID 7
and the container ID to 0.
*** /Jasmine jasminelv1 jasminevg ***
Written to the map_file for Tablespace ID 7.
Finished processing the Tablespace ID 7.
Finding the mount point for Tablespace ID 8.
Getting the path for Tablespace ID 8

verifying the path with the mount command for Tablespace ID 8
and the container ID to 0.
*** /Jasmine jasminelv1 jasminevg ***
Written to the map_file for Tablespace ID 8.
Finished processing the Tablespace ID 8.
Finding the mount point for Tablespace ID 9.
Getting the path for Tablespace ID 9
verifying the path with the mount command for Tablespace ID 9
and the container ID to 0.
*** /Jasmine1 jasminelv2 jasminevg ***
Written to the map_file for Tablespace ID 9.
Finished processing the Tablespace ID 9.
Finding the mount point for Tablespace ID 10.
Getting the path for Tablespace ID 10
verifying the path with the mount command for Tablespace ID 10
and the container ID to 0.
*** /Jasmine1 jasminelv2 jasminevg ***
Written to the map_file for Tablespace ID 10.
Finished processing the Tablespace ID 10.
Finding the mount point for Tablespace ID 11.
Getting the path for Tablespace ID 11
verifying the path with the mount command for Tablespace ID 11
and the container ID to 0.
*** RAWVOLUME /dev/rLilylv Lilylv Lilyvg ***
Written to the map_file for Tablespace ID 11.
Finished processing the Tablespace ID 11.
Disconnecting from the database flowers
Disconnected from the database flowers.
Copying the Mapfile for restore to the host pseries1
Copying the file : VG025043418.lv
Copying the file : VG025043418.allvol
Copying the file : VG125043418.lv
Copying the file : VG125043418.allvol
Copying the file : VG225043418.lv
Copying the file : VG225043418.allvol
Copying the file : VG325043418.lv
Copying the file : VG325043418.allvol
Done the query command successfully.
A Flashcopy volume does not exist for the volume "Camelia". Creating one...
The VolumeCopy volume "VC200502250434180" has been successfully configured for backing up.
A Flashcopy volume does not exist for the volume "Flora". Creating one...

The VolumeCopy volume "VC200502250434181" has been successfully configured for backing up.

A Flashcopy volume does not exist for the volume "Hana". Creating one...

The VolumeCopy volume "VC200502250434182" has been successfully configured for backing up.

A Flashcopy volume does not exist for the volume "Michie". Creating one...

The VolumeCopy volume "VC200502250434183" has been successfully configured for backing up.

A Flashcopy volume does not exist for the volume "Jasmine". Creating one...

The VolumeCopy volume "VC200502250434184" has been successfully configured for backing up.

A Flashcopy volume does not exist for the volume "Lily". Creating one...

The VolumeCopy volume "VC200502250434185" has been successfully configured for backing up.

Completed checking for the optimal state of the necessary storage components. Proceeding with backup...

Connect to DBhost for flashcopy database succeeded

Connecting to Database. flowers

Connected successfully to database. flowers

Executing the SMdevices command on localhost.

Finding the mount point for Database directory.

*** /Camelia Camelialv Cameliavg ***

Finished processing the Database directory

Finding the mount point for Tablespace ID 0.

Getting the path for Tablespace ID 0

verifying the path with the mount command for Tablespace ID 0

and the container ID to 0.

*** /Camelia Camelialv Cameliavg ***

Written to the map_file for Tablespace ID 0.

Finished processing the Tablespace ID 0.

Finding the mount point for Tablespace ID 1.

Getting the path for Tablespace ID 1

verifying the path with the mount command for Tablespace ID 1

and the container ID to 0.

*** /Camelia Camelialv Cameliavg ***

Written to the map_file for Tablespace ID 1.

Finished processing the Tablespace ID 1.

Finding the mount point for Tablespace ID 2.

Getting the path for Tablespace ID 2

verifying the path with the mount command for Tablespace ID 2

and the container ID to 0.

*** /Camelia Camelialv Cameliavg ***

Written to the map_file for Tablespace ID 2.

Finished processing the Tablespace ID 2.

Finding the mount point for Tablespace ID 3.
Getting the path for Tablespace ID 3
verifying the path with the mount command for Tablespace ID 3
and the container ID to 0.
*** /Camelia Camelialv Cameliavg ***
Written to the map_file for Tablespace ID 3.
Finished processing the Tablespace ID 3.
Finding the mount point for Tablespace ID 4.
Getting the path for Tablespace ID 4
verifying the path with the mount command for Tablespace ID 4
and the container ID to 0.
*** /Camelia Camelialv Cameliavg ***
Written to the map_file for Tablespace ID 4.
Finished processing the Tablespace ID 4.
Finding the mount point for Tablespace ID 5.
Getting the path for Tablespace ID 5
verifying the path with the mount command for Tablespace ID 5
and the container ID to 0.
*** /Michie fslv01 Michievg ***
Written to the map_file for Tablespace ID 5.
Finished processing the Tablespace ID 5.
Finding the mount point for Tablespace ID 6.
Getting the path for Tablespace ID 6
verifying the path with the mount command for Tablespace ID 6
and the container ID to 0.
*** /Michie fslv01 Michievg ***
Written to the map_file for Tablespace ID 6.
Finished processing the Tablespace ID 6.
Finding the mount point for Tablespace ID 7.
Getting the path for Tablespace ID 7
verifying the path with the mount command for Tablespace ID 7
and the container ID to 0.
*** /Jasmine jasminelv1 jasminevg ***
Written to the map_file for Tablespace ID 7.
Finished processing the Tablespace ID 7.
Finding the mount point for Tablespace ID 8.
Getting the path for Tablespace ID 8
verifying the path with the mount command for Tablespace ID 8
and the container ID to 0.
*** /Jasmine jasminelv1 jasminevg ***
Written to the map_file for Tablespace ID 8.
Finished processing the Tablespace ID 8.

```
Finding the mount point for Tablespace ID 9.
Getting the path for Tablespace ID 9
verifying the path with the mount command for Tablespace ID 9
and the container ID to 0.
*** /Jasmine1 jasminelv2 jasminevg ***
Written to the map_file for Tablespace ID 9.
Finished processing the Tablespace ID 9.
Finding the mount point for Tablespace ID 10.
Getting the path for Tablespace ID 10
verifying the path with the mount command for Tablespace ID 10
and the container ID to 0.
*** /Jasmine1 jasminelv2 jasminevg ***
Written to the map_file for Tablespace ID 10.
Finished processing the Tablespace ID 10.
Finding the mount point for Tablespace ID 11.
Getting the path for Tablespace ID 11
verifying the path with the mount command for Tablespace ID 11
and the container ID to 0.
*** RAWVOLUME /dev/rLilylv Lilylv Lilyvg ***
Written to the map_file for Tablespace ID 11.
Finished processing the Tablespace ID 11.
Disconnecting from the database flowers
Disconnected from the database flowers.
Copying the Mapfile for restore to the host pseries1
Copying the file : VG025043418.lv
Copying the file : VG025043418.allvol
Copying the file : VG125043418.lv
Copying the file : VG125043418.allvol
Copying the file : VG225043418.lv
Copying the file : VG225043418.allvol
Copying the file : VG325043418.lv
Copying the file : VG325043418.allvol
Done the query command successfully.
Connecting to Database. flowers
Connected successfully to database. flowers
Suspending writes to the database flowers
Suspending writes to the database done successfully.
Unquiescing the database flowers
Unquiescing of the database flowers successful
Committing the database flowers before disconnect.
Disconnecting from the database flowers
Disconnected from the database: flowers.
```

Checking the status of the VolumeCopy operation...

Successfully copied the volume "Jasmine-SNBkup" onto the volume "VC200502250434184" using VolumeCopy.

Successfully copied the volume "Lily-SNBkup" onto the volume "VC200502250434185" using VolumeCopy.

Successfully copied the volume "Camelia-SNBkup" onto the volume "VC200502250434180" using VolumeCopy.

Successfully copied the volume "Flora-SNBkup" onto the volume "VC200502250434181" using VolumeCopy.

Successfully copied the volume "Hana-SNBkup" onto the volume "VC200502250434182" using VolumeCopy.

Successfully copied the volume "Michie-SNBkup" onto the volume "VC200502250434183" using VolumeCopy.

Successfully completed the VolumeCopy operation for all the volumes.

Recreating the volume group VG025043418...

Recreating the volume group VG125043418...

Recreating the volume group VG225043418...

Recreating the volume group VG325043418...

Completed configuring this backup host for backing up the database flowers.

Checking file system consistency for the volumes of the database "flowers"...

Successfully replayed the logs for the logical volume "LV025043418".

Successfully completed file consistency check for the logical volume "LV025043418".

Successfully replayed the logs for the logical volume "LV325043418".

Successfully completed file consistency check for the logical volume "LV325043418".

Successfully replayed the logs for the logical volume "LV425043418".

Successfully completed file consistency check for the logical volume "LV425043418".

Successfully replayed the logs for the logical volume "LV525043418".

Successfully completed file consistency check for the logical volume "LV525043418".

The logical volume "LV725043418" is a raw volume without a file system.

File system consistency check successfully completed for all volumes of the database "flowers".

Restore Mapfile Contents

Backup performed on Feb-25-2005 at 04:34:56 AM.

Database: flowers in /Camelia/db21inst/NODE0000/SQL00001/

Database instance name: db21inst

Storage Volume label: Camelia

Storage Subsystem name: Indi57

Database instance name: db21inst

Storage Volume label: Flora

Storage Subsystem name: Indi57

Database instance name: db21inst

Storage Volume label: Hana

Storage Subsystem name: Indi57
Database Path: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 0 is /Camelia/db21inst/NODE0000/SQL00001/SQLT0000.0
Tablespace ID 0: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 1 is /Camelia/db21inst/NODE0000/SQL00001/SQLT0001.0
Tablespace ID 1: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 2 is /Camelia/db21inst/NODE0000/SQL00001/SQLT0002.0
Tablespace ID 2: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 3 is /Camelia/t1
Tablespace ID 3: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 4 is /Camelia/t2
Tablespace ID 4: /Camelia Camelialv Cameliavg LV025043418

TABLE SPACE DETAILS:

Dir path for table space 5 is /Michie/t3
Database instance name: db21inst
Storage Volume label: Michie
Storage Subsystem name: Indi57
Tablespace ID 5: /Michie fslv01 Michievg LV325043418

TABLE SPACE DETAILS:

Dir path for table space 6 is /Michie/t4
Tablespace ID 6: /Michie fslv01 Michievg LV325043418

TABLE SPACE DETAILS:

Dir path for table space 7 is /Jasmine/t5
Database instance name: db21inst
Storage Volume label: Jasmine
Storage Subsystem name: Arjuna
Tablespace ID 7: /Jasmine jasminelv1 jasminevg LV425043418

TABLE SPACE DETAILS:

Dir path for table space 8 is /Jasmine/t6
Tablespace ID 8: /Jasmine jasminelv1 jasminevg LV425043418

TABLE SPACE DETAILS:

Dir path for table space 9 is /Jasmine1/t7
Tablespace ID 9: /Jasmine1 jasminelv2 jasminevg LV525043418

TABLE SPACE DETAILS:

Dir path for table space 10 is /Jasmine1/t8
Tablespace ID 10: /Jasmine1 jasminelv2 jasminevg LV525043418

TABLE SPACE DETAILS:

Dir path for table space 11 is /dev/rLilylv
Database instance name: db21inst
Storage Volume label: Lily
Storage Subsystem name: Arjuna
Tablespace ID 11: /dev/rLilylv Lilylv Lilyvg LV725043418

VG details for Cameliavg
Cameliavg:

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
Camelialv	jfs	1000	1000	2	open/syncd	/Camelia
loglv13	jfslog	1	1	1	open/syncd	N/A

VG details for Michievg
Michievg:

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
loglv14	jfs2log	1	1	1	open/syncd	N/A
fslv01	jfs2	512	512	1	open/syncd	/Michie

VG details for jasminevg
jasminevg:

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
jasminelv1	jfs	300	300	1	open/syncd	/Jasmine
jasminelv2	jfs	300	300	1	open/syncd	/Jasmine1
loglv15	jfslog	1	1	1	open/syncd	N/A

VG details for Lilyvg
Lilyvg:

LV NAME	TYPE	LPs	PPs	PVs	LV STATE	MOUNT POINT
Lilylv	jfs	630	630	1	closed/syncd	N/A

Mapping Information for VolumeCopy volumes:

Storage Volume Label: Camelia
VolumeCopy Volume Label: VC200502250434180
Storage subsystem: Indi57

Storage Volume Label: Flora
VolumeCopy Volume Label: VC200502250434181
Storage subsystem: Indi57

Storage Volume Label: Hana
VolumeCopy Volume Label: VC200502250434182
Storage subsystem: Indi57

Storage Volume Label: Michie
VolumeCopy Volume Label: VC200502250434183
Storage subsystem: Indi57

Storage Volume Label: Jasmine
VolumeCopy Volume Label: VC200502250434184
Storage subsystem: Arjuna

Storage Volume Label: Lily
VolumeCopy Volume Label: VC200502250434185
Storage subsystem: Arjuna

*** END OF RESTORE MAP FILE ***

Successfully backed up all the volumes of the database "flowers."

Unmapping the VolumeCopy destination volumes from the backup server...
Completed unmapping the VolumeCopy destination volumes from the backup server.
Disabling Flashcopy Volumes...
hdisk20 deleted
hdisk21 deleted
hdisk22 deleted
hdisk23 deleted
hdisk18 deleted
hdisk19 deleted
Disabled all Flashcopy Volumes.
Ready to backup the next database...

All databases backed up successfully.
The debug option is set. Therefore, manually delete the temp files in /rajesh/feb17/db_bkup_app_2.0/.internal/.tempFiles_20050225043418 later.
This backup operation was completed at 05:29:57 AM on Feb-25-2005.

Glossary

General Terms

AIX Failover Driver
Software that allows redundant IO paths from a server to a DS4000 to be used. If an I/O path fails, all I/Os are sent across the surviving path. AIX failover driver supports 32 LUNS.

Array
A group of related volumes that can be managed as an entity by the DS4000 Manager. A set of drives that the DS4000 logically groups together to provide one or more volumes to a host server. To create an array, two parameters are specified: a RAID level, and capacity.

Base Directory
An IBD term that identifies a directory, other than the root directory, in which to store the IBD scripts.

CRON
An abbreviation for chronological; cron is a readily available UNIX utility that is used to schedule application programs and processes.

Database
A set of related database objects such as tables, views and indexes. A relational database consists of one or more logical units called table spaces.

Database 2 (DB2)
DB2 is a relational database management system supplied by IBM.

Database Control Files
Some DBMS maintain control files to hold additional information about the physical structure of a database, such as which physical files are used by each table space.

Database Instance
An instantiation of a Database Manager. For example, when DB2 is installed on a system, program files for the DB2 Database Manager are physically copied to a specific location on that machine and one instance of the DB2 Database Manager is created. Additional instances of the DB2 Database Manager can be created for a particular system. Even though DB2 Database Managers share physical code, each instance behaves like a separate installation of DB2.

Database Table
A uniquely identified unit of storage for a relational database management system (RDBMS) that is maintained within a table space. Each table is a logical structure that is used to present data as a collection of rows within a fixed number of columns.

Database Table Space
Tables and indexes are assigned to table spaces; one or more tables or indexes can be assigned to a particular table space. Table spaces are logical entities that provide a convenient way of separating the user's view of data from considerations associated with storing data on disks.

DS4000
A series of highly available and high performance storage servers (supplied by IBM) that implement RAID technology.

DS4000 Partitions (Host Partitions)
A partition is a logical entity consisting of one or more volumes arranged in arrays. Partitions enable host servers to share access to storage volumes by granting the servers exclusive or shared access to a partition. Partitions are a DS4000 premium feature.

DS4000 Storage Manager
A graphical management interface that controls, monitors, and manages DS4000's.

FlashCopy
A feature of a DS4000 that creates point-in-time virtual copies of volumes, providing an instantaneous copy of a DS4000 volume as it existed at the point-in-time when FlashCopy was invoked. Two logical volumes exist in a FlashCopy relationship: a source volume and a target volume.

FlashCopy Source Volume
Source volume in a FlashCopy relationship is a production volume that contains the data that is copied to another volume, which is known as a FlashCopy target volume.

FlashCopy Target Volume
Target volume in a FlashCopy relationship is a virtual image of the source volume at the time when the FlashCopy operation was invoked. As data is changed on a source volume, the “before state” of the data blocks are copied to the target volume.

High Availability Cluster Multi-Processing
An IBM software product generally referred to as HACMP. HACMP provides clustering services for to increase application availability and scalability.

Hot_Add
A DS4000 Storage Manager utility that actively scans for new volumes or LUNs in a DS4000.

JFS
An IBM software product named Journal File System, a 32-bit native file system on AIX servers. AIX uses database journaling techniques to maintain structural consistency and prevent damage to the file system when a system is halted abnormally.

JFS 2
An IBM software product, a 64-bit version of the Journal File System.

Logical Volume Manager
A utility that allows logical volumes to span multiple physical volumes. Data on logical volumes appears to be contiguous to the user, but might not be contiguous on the physical volume.

Logical Unit Number (LUN)
The number a host server uses to access a volume of a DS4000. Each host server has its own LUN address space and, therefore, the same LUN can be used by different host servers to access different volumes on a DS4000.

Physical Volume Name
The name (hdisk) of any of the physical volumes contained in an array (AIX OS).

RAID
An abbreviation for redundant array of independent disks. A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

Raw I/O
A method of performing input and output (I/O) in UNIX in which an application handles the I/O control. Raw I/O is sometimes referred to as unformatted I/O.

Redo Log
A set of files that record all changes made to a database so that, in the event of a failure, updates to the database are not lost. The Redo Log typically consists of two parts: an online redo log and an archive log.

SATA Drives
Hard disks that implement a serial ATA protocol and cost less than Fibre Channel disk drives of comparable capacity.

Set Write Suspend
A DB2 command that suspends writes to a database.

Set Write Resume
A DB2 command that resumes writes to a database.

SMCli
A command line interface (CLI) provided with the DS4000 Manager.

SMdevices
A DS4000 Manager utility that is used to associate the physical device name with the volume name that is used by the DS4000 Manager. SMdevices is used by the IBD Solution.

Split-Mirror Backup
An online database backup technique that is supported in DB2 and is used to free a production system from the overhead of a backup. An independent copy or mirror of a database is created and that copy is used for backup.

Standard Volume
A DS4000-specific term used to describe a logical structure created on a DS4000 for data storage. A standard volume is defined over a set of drives called an array, and has a defined RAID level and capacity.

TSM
An abbreviation for Tivoli Storage Manager.

VCvolume
An abbreviation for VolumeCopy volume.

Volume
A volume is a logical unit of storage that can be addressed by an operating system.

Volumegroup Name
A system-wide unique name assigned to a Volume group when it is imported into a system (from the perspective of the Logical Volume Manager).

Volume Label
A name used by an operating system to address a Volume.

IBD-Specific Terms

Association
A group of one or more database servers and a recovery sever where the recovery server provides backup services to the database servers in the association.

Backup Cycle
IBD backups the databases identified in a Configuration File sequentially. The process of backing-up each individual database is referred to as a Backup Cycle.

Configuration File
The administrative interface to IBD. One or more databases within a DB2 instance are identified for backup within a Configuration File and a backup policy for those databases is defined.

Db_bkup_app_2.x.tar
A tar file containing the IBD software, readme, release notes and user guide.

DS4000 Non-Optimal
The abnormal operating state, in which the storage subsystem is operating in a degraded mode, such as when a disk rebuild operation is in process.

DS4000 Optimal
The normal operating state, in which all storage subsystem components, including attached disks, are properly functioning.

Fast_backup.sh
IBD script used to initiate one or more backup cycles. Fast_backup.sh can be invoked manually at the command line or attached to a cron to run automatically on a user-define cycle. Fast_backup.sh is run with a Configuration File as input.

FCvolume
A virtual point-in-time image of a database created by IBD during a backup cycle by invoking a FlashCopy operation. Also referred to as a FlashCopy volume.

IBD
A solution provided by IBM that non-disruptively backs-up online databases using FlashCopy and optionally, VolumeCopy technology.

IBD Log
A file generated by IBD during each backup cycle that contains information on each IBD backup cycle, including errors and events.

Install_on_DB_host.sh
A self-extracting and self-install script used to install IBD on one or more database servers in a single execution.

Quick Recovery Volume
A VCvolume initialized as a database by DB2 and used to quickly recovery a corrupt database. After initialization, the QRV can be rolled forward to the point-in-time that the corruption occurred.

Regular Backup
IBD supports two types of backup cycles: regular and resumed. A regular backup is the normal operating state. See resumed backup.

Resumed Backup
IBD supports two types of backup cycles: regular and resumed. A resume backup can be used after a failed backup cycle to restart the backup cycle from the point of failure (VolumeCopy operation or a TSM operation) and attempt to complete it. If a resumed backup completes, the database will be backed up as of the point-in-time that FlashCopy was invoked.

Restore_Mapfile
Metadata file created by IBD during each backup cycle and is backed up along with the database by TSM in the location: *MOUNT_POINT_HEAD specified in the configuration file* / *DB_INSTANCE specified in the configuration file* / *Database Name* / Restore Mapfile. If the backup policy does not include off-loading databases to tape, the Restore_Mapfile is appended to the backup cycle log file and stored on the recovery server.

VCvolume
A physical point-in-time image of a database that is optionally created by IBD during a backup cycle by invoking a VolumeCopy operation and copying an FCvolume. Also referred to as a VolumeCopy volume.

Readers' Comments — We'd Like to Hear from You

IBM TotalStorage DS4000 Integrated Backup for Databases (with DB2)
User Guide

Publication No. GA32-0474-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

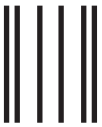
Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GA32-0474-00

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Information Development
Department GZW
9000 South Rita Road
Tucson, Arizona U.S.A. 85775-4401

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Printed in USA

GA32-0474-00