DB2 for z/OS Using EMC Symmetrix Storage Systems

Version 2.0

- System and Object Cloning
- Backup, Restore, and Recovery
- Layout and Performance

Paul Pendle
## Preface

### Chapter 1  An Introduction to DB2 for z/OS

Overview ................................................................................................................. 20  
  DB2 system elements .................................................................................. 20  
  DB2 data elements ................................................................................... 22  
  DB2 data sharing ..................................................................................... 23  
  DB2 attachment facilities ..................................................................... 24  
Using storage management for managing DB2 systems ...................... 26  
  Cloning DB2 environments and objects ........................................... 26  
  Backup and recovery ........................................................................... 26

### Chapter 2  EMC Foundation Products

Introduction ............................................................................................................. 28  
Symmetrix hardware and EMC Enginuity features .......................... 30  
  Symmetrix VMAX platform ................................................................. 30  
  EMC Enginuity operating environment .......................................... 32  
  Symmetrix features for mainframe ..................................................... 33  
ResourcePak Base for z/OS ................................................................. 36  
  Features .................................................................................................. 37  
SRDF family of products for z/OS .............................................................. 41  
  SRDF Host Component for z/OS ......................................................... 42  
  SRDF mainframe features ................................................................ 42  
  Concurrent SRDF and SRDF/Star ...................................................... 43  
  Multi-Session Consistency ................................................................ 45  
  SRDF/AR ............................................................................................. 45  
  EMC Geographically Dispersed Disaster Restart ............................ 45
# Contents

SRDF Enterprise Consistency Group for z/OS ...................... 46
Restart in the event of a disaster or nondisaster ............... 52
EMC AutoSwap ................................................................. 53
AutoSwap highlights ....................................................... 54
Use cases ........................................................................ 55
EMC TimeFinder family products for z/OS ...................... 56
  TimeFinder/Clone for z/OS ............................................. 56
  TimeFinder/Snap for z/OS .............................................. 57
  TimeFinder/Mirror for z/OS .......................................... 58
  TimeFinder/CG ............................................................. 59
Consistent dataset snap .................................................... 60
EMC z/OS Storage Manager .............................................. 61
Symmetrix Management Console .................................. 62
Symmetrix Performance Analyzer ................................ 63
Unisphere for VMAX ....................................................... 64
Virtual Provisioning ......................................................... 65
Fully Automated Storage Tiering .................................. 65
Data at Rest Encryption .................................................. 66

## Chapter 3  Advanced Storage Provisioning

Virtual Provisioning .......................................................... 68
  Terminology .................................................................. 68
  Overview ................................................................. 70
  Virtual Provisioning operations .................................... 71
  Host allocation considerations .................................... 73
  Balanced configurations ............................................. 73
  Pool management ....................................................... 74
  Thin device monitoring .............................................. 76
  Considerations for DB2 components on thin devices ...... 76
  Hash table spaces ...................................................... 77
  Replication considerations ........................................ 78
  Performance considerations ........................................ 80

Fully Automated Storage Tiering .................................. 82
  Introduction ............................................................. 82
  Best practices for DB2 and FAST VP ......................... 84
  Unisphere for VMAX ................................................... 85
  DB2 and SMS storage groups ..................................... 88
  DB2 and HSM .......................................................... 88

## Chapter 4  Cloning of DB2 Objects and Systems

DB2 object cloning ........................................................ 92
Creating objects on the target DB2 ...........................................94
Cloning the data ........................................................................95
Resetting the level ID on the target ........................................104
Matching the DBID/PSID to the target catalog .....................105
DB2 object cloning considerations ......................................106
Cloning method usage considerations ..................................108
DB2 system cloning ...............................................................110
Cloning the DB2 system within the same LPAR .................111
Cloning a DB2 system to another LPAR ..............................123
Recoverability of the cloned target DB2 system .................128

Chapter 5  
Back up DB2 Environments

Overview .................................................................................. 130
Recoverable vs. restartable copies of databases ................... 131
  Recoverable disk copies .....................................................131
  Restartable disk copies .....................................................131
Database organization to facilitate recovery ....................... 132
  Preparing for an array-based backup .................................133
EMC replication for DB2 system backup .............................. 135
  Preparing the array for replication ..................................135
  Making a copy of the DB2 volumes .................................138
  Resetting the source system ............................................140
  Presenting the target volumes to a host LPAR .................141
  Backing up the target volumes .......................................142
Creating a remote backup ...................................................... 144
Backup method considerations .............................................. 147
Creating multiple array copies ............................................ 147
Reporting on dataset placement on backup volumes ........... 149
DB2 image copy considerations .......................................... 150
Considerations for 32 KB pages .......................................... 150
SYSLGRNX and SYSCOPY recovery .................................... 151
Operations with LOG NO considerations .......................... 151
Backing up DB2 objects with dataset snap ......................... 151
Backup method comparison ................................................. 154
Symmetrix arrays with DB2 system point-in-time recovery .... 155
  System point-in-time recovery overview ..........................155
  Compatible Flash for z/OS and DB2 .................................157
  Compatible Native Flash for Mainframe ............................158
  SYPITR with TimeFinder .................................................158
Backing up DB2 V9 for object-level recovery ....................... 158
DB2 V10 FlashCopy Image Copy ......................................... 159
Space Efficient FlashCopy ................................................. 160
## Contents

Rocket Backup and Recovery for DB2 on z/OS.......................... 161

### Chapter 6  DB2 Recovery Procedures

- Overview.................................................................................. 164
- TimeFinder restore and parallel recovery.............................. 165
- DB2 system restart/recovery.................................................. 165
  - DB2 system restore to copy.................................................. 166
  - DB2 system restore and roll-forward .................................. 167
- Optimizing DB2 recovery....................................................... 169
- Recovering DB2 objects.......................................................... 172
  - Object restore from system backup................................. 172
  - Restoring objects using TimeFinder/Snap......................... 172
  - Full-volume restore for database objects......................... 173
  - Restoring objects from tape............................................. 174
- DB2 System Point-in-Time Recovery tools............................ 176
  - EMC Compatible Flash for z/OS.................................... 177
  - EMC Compatible Native Flash for Mainframe .................. 177
  - Combining SYSPITR with TimeFinder............................... 177
- Rocket Backup and Recovery for z/OS................................... 178
- DB2 steps for recovery in detail............................................ 179
  - Shutting down the DB2 system........................................ 179
  - Vary off the DB2 source volumes.................................. 179
  - TimeFinder system restore function.............................. 180
  - System restore from tape............................................... 182
  - Vary on the DB2 source volumes................................. 183
  - Create a conditional restart record............................... 184
  - Bring up DB2 in DEFER ALL mode................................. 184
  - Using SYSPITR to restore the DB2 system....................... 185
  - Define recatalog of a DB2 dataset................................. 185
  - Dataset recover logonly............................................... 186

### Chapter 7  Understanding Disaster Restart and Disaster Recovery

- Overview.................................................................................. 188
- Disaster recovery vs. disaster restart................................. 189
- Definitions................................................................................ 190
  - Dependent-write consistency........................................... 190
  - Database restart.............................................................. 191
  - Database recovery............................................................ 191
  - Forward recovery............................................................. 192
- Design factors for disaster recovery and disaster restart..... 192
  - Recovery point objective................................................ 193
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer pool I/O</td>
<td>264</td>
</tr>
<tr>
<td>Active log I/O</td>
<td>264</td>
</tr>
<tr>
<td>Bandwidth reduction</td>
<td>265</td>
</tr>
<tr>
<td>Temp space replication</td>
<td>265</td>
</tr>
<tr>
<td>DB2 use of PAVs on Symmetrix arrays</td>
<td>265</td>
</tr>
<tr>
<td>Extended address volumes</td>
<td>266</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Chapter 9 SAP Environments on z/OS</td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td>270</td>
</tr>
<tr>
<td>DB2 use in an SAP environment</td>
<td>271</td>
</tr>
<tr>
<td>Storage management</td>
<td>271</td>
</tr>
<tr>
<td>Cloning SAP environments</td>
<td>271</td>
</tr>
<tr>
<td>Backup and recovery</td>
<td>272</td>
</tr>
<tr>
<td>Holistic business continuity for SAP</td>
<td>272</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Appendix A Related Documents</td>
<td></td>
</tr>
<tr>
<td>Related documents</td>
<td>276</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Appendix B REXX Sample Scripts</td>
<td></td>
</tr>
<tr>
<td>Overview</td>
<td>278</td>
</tr>
<tr>
<td>REXX Example #1</td>
<td>279</td>
</tr>
<tr>
<td>REXX Example #2</td>
<td>286</td>
</tr>
<tr>
<td>REXX Example #3</td>
<td>289</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Glossary</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
</tr>
</tbody>
</table>
DB2 for z/OS Using EMC Symmetrix Storage Systems Version 2.0

Figures

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 DB2 started tasks</td>
<td>21</td>
</tr>
<tr>
<td>2 DB2 data elements: Non-data sharing</td>
<td>22</td>
</tr>
<tr>
<td>3 DB2 data sharing system and data elements</td>
<td>24</td>
</tr>
<tr>
<td>4 DB2 attachment facilities</td>
<td>25</td>
</tr>
<tr>
<td>5 Symmetrix hardware and software layers</td>
<td>30</td>
</tr>
<tr>
<td>6 Symmetrix VMAX logical diagram</td>
<td>32</td>
</tr>
<tr>
<td>7 z/OS SymmAPI architecture</td>
<td>36</td>
</tr>
<tr>
<td>8 SRDF family for z/OS</td>
<td>42</td>
</tr>
<tr>
<td>9 Classic SRDF/Star support configuration</td>
<td>44</td>
</tr>
<tr>
<td>10 SRDF Consistency Group using SRDF-ECA</td>
<td>50</td>
</tr>
<tr>
<td>11 DB2 ConGroup setup</td>
<td>52</td>
</tr>
<tr>
<td>12 AutoSwap before and after states</td>
<td>54</td>
</tr>
<tr>
<td>13 TimeFinder family of products for z/OS</td>
<td>56</td>
</tr>
<tr>
<td>14 EMC z/OS Storage Manager functionality</td>
<td>61</td>
</tr>
<tr>
<td>15 Thin devices, pools, and physical disks</td>
<td>72</td>
</tr>
<tr>
<td>16 Thin device pool with devices filling up</td>
<td>75</td>
</tr>
<tr>
<td>17 Thin device pool rebalanced with new devices</td>
<td>75</td>
</tr>
<tr>
<td>18 FAST VP storage groups, policies, and tiers</td>
<td>84</td>
</tr>
<tr>
<td>19 DB2 object cloning overview</td>
<td>93</td>
</tr>
<tr>
<td>20 Cloning DB2 objects using TimeFinder dataset snap</td>
<td>96</td>
</tr>
<tr>
<td>21 Object cloning using TimeFinder/Mirror</td>
<td>98</td>
</tr>
<tr>
<td>22 Cloning DB2 objects using TimeFinder full volume snap</td>
<td>102</td>
</tr>
<tr>
<td>23 Object cloning method decision process</td>
<td>109</td>
</tr>
<tr>
<td>24 DB2 system cloning</td>
<td>110</td>
</tr>
<tr>
<td>25 Cloning—data sharing to non-data sharing on the same LPAR</td>
<td>116</td>
</tr>
<tr>
<td>26 Cloning—data sharing to non-data sharing on a different LPAR</td>
<td>126</td>
</tr>
<tr>
<td>27 Recommended dataset placement</td>
<td>133</td>
</tr>
<tr>
<td>28 Remote backup using TimeFinder and SRDF</td>
<td>144</td>
</tr>
<tr>
<td>29 DB2 configured with three sets of target volumes</td>
<td>148</td>
</tr>
<tr>
<td>30 DB2 object backup using dataset snap</td>
<td>152</td>
</tr>
<tr>
<td>Page</td>
<td>Topic</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>31</td>
<td>DB2 object backup using volume-based TimeFinder copies</td>
</tr>
<tr>
<td>32</td>
<td>DB2 BACKUP SYSTEM FULL (default)</td>
</tr>
<tr>
<td>33</td>
<td>DB2 BACKUP SYSTEM DATAONLY</td>
</tr>
<tr>
<td>34</td>
<td>DB2 system restore to copy</td>
</tr>
<tr>
<td>35</td>
<td>Restore of system data structures (no log or BSDS)</td>
</tr>
<tr>
<td>36</td>
<td>Single Symmetrix array to single Symmetrix array replication</td>
</tr>
<tr>
<td>37</td>
<td>Rolling disaster with multiple production Symmetrix arrays</td>
</tr>
<tr>
<td>38</td>
<td>Rolling disaster with SRDF Consistency Group protection</td>
</tr>
<tr>
<td>39</td>
<td>SRDF/S—Multiple source Symmetrix arrays and ConGroup protection...</td>
</tr>
<tr>
<td>40</td>
<td>DB2 ConGroup configuration</td>
</tr>
<tr>
<td>41</td>
<td>SRDF/A replication configuration</td>
</tr>
<tr>
<td>42</td>
<td>SRDF/AR single-hop replication configuration</td>
</tr>
<tr>
<td>43</td>
<td>SRDF/AR multi-hop replication configuration</td>
</tr>
<tr>
<td>44</td>
<td>SRDF/Star configuration</td>
</tr>
<tr>
<td>45</td>
<td>SRDF/EDP block diagram</td>
</tr>
<tr>
<td>46</td>
<td>GDDR with SRDF/Star and AutoSwap</td>
</tr>
<tr>
<td>47</td>
<td>EMC foundation products for GDDR</td>
</tr>
<tr>
<td>48</td>
<td>The performance stack</td>
</tr>
<tr>
<td>49</td>
<td>Relationship between host block size and IOPS/throughput</td>
</tr>
<tr>
<td>50</td>
<td>RAID 5 (3+1) layout detail</td>
</tr>
<tr>
<td>51</td>
<td>Anatomy of a RAID 5 random write</td>
</tr>
<tr>
<td>52</td>
<td>Magnetic disk capacity and performance history</td>
</tr>
<tr>
<td>53</td>
<td>Table partitioning on tiered storage</td>
</tr>
<tr>
<td>54</td>
<td>Disk performance factors</td>
</tr>
<tr>
<td>55</td>
<td>VSAM striping for DB2 active logs</td>
</tr>
<tr>
<td>56</td>
<td>High-level SAP processing environment</td>
</tr>
</tbody>
</table>
Tables

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Virtual Provisioning terms</td>
<td>68</td>
</tr>
<tr>
<td>2 Thick-to-thin host migrations</td>
<td>79</td>
</tr>
<tr>
<td>3 Thick-to-thin VMAX array-based migrations</td>
<td>80</td>
</tr>
<tr>
<td>4 DB2 access types allowed when cloning data between DB2 systems</td>
<td>107</td>
</tr>
<tr>
<td>5 TimeFinder backup methods comparison</td>
<td>154</td>
</tr>
<tr>
<td>6 Recovery considerations</td>
<td>164</td>
</tr>
</tbody>
</table>
As part of an effort to improve and enhance the performance and capabilities of its product lines, EMC periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, please contact your EMC representative.

Note: This document was accurate as of the time of publication. However, as information is added, new versions of this document may be released to EMC Online Support. Check EMC Online Support website to ensure that you are using the latest version of this document.

Purpose
This document provides a comprehensive treatment of EMC Symmetrix technologies that can be used to enhance DB2 for z/OS databases.

Audience
This TechBook is part of the Symmetrix documentation set and is intended for use by database and system administrators, systems integrators, and members of EMC Technical Global Services that support and work with DB2 for z/OS.

Readers of this document are expected to be familiar with the following topics:

- DB2 for z/OS
- z/OS operating system
- Symmetrix fundamentals
Organization

This TechBook is divided into the following eight chapters and appendices:

Chapter 1, “An Introduction to DB2 for z/OS,” provides a high-level overview of DB2 for z/OS and its operating environment.

Chapter 2, “EMC Foundation Products,” describes EMC products used to support the management of DB2 for z/OS environments.

Chapter 4, “Cloning of DB2 Objects and Systems,” describes procedures to clone DB2 systems and DB2 objects from one DB2 to another.

Chapter 5, “Backing Up DB2 Environments,” describes how to back up DB2 environments using EMC Replication Technology.

Chapter 6, “DB2 Recovery Procedures,” describes how to restore and recover DB2 systems and objects.

Chapter 7, “Understanding Disaster Restart and Disaster Recovery,” describes the difference between using traditional recovery techniques versus EMC restart solutions.

Chapter 8, “Performance Topics,” describes best practices for DB2 performance on Symmetrix subsystems.

Chapter 9, “SAP Environments on z/OS,” describes the special considerations when deploying SAP using DB2 for z/OS with Symmetrix subsystems.

Appendix A provides a list of references to additional material available from IBM and EMC.

Appendix B gives example REXX scripts for cloning activities. The scripts that are provided in this document cover methods for performing various DB2 functions using Symmetrix subsystems with EMC software. These examples were developed in laboratory testing and may need tailoring to suit other operational environments. Any procedures outlined in this document should be thoroughly tested before production implementation.

Conventions used in this document

EMC uses the following conventions for special notices:
Typographical conventions
EMC uses the following type style conventions in this document.

Normal
Used in running (nonprocedural) text for:
- Names of interface elements (such as names of windows, dialog boxes, buttons, fields, and menus)
- Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, functions, utilities
- URLs, pathnames, filenames, directory names, computer names, filenames, links, groups, service keys, file systems, notifications

Bold
Used in running (nonprocedural) text for:
- Names of commands, daemons, options, programs, processes, services, applications, utilities, kernels, notifications, system calls, man pages

Used in procedures for:
- Names of interface elements (such as names of windows, dialog boxes, buttons, fields, and menus)
- What user specifically selects, clicks, presses, or types

Italic
Used in all text (including procedures) for:
- Full titles of publications referenced in text
- Emphasis (for example a new term)
- Variables

Courier
Used for:
- System output, such as an error message or script
- URLs, complete paths, filenames, prompts, and syntax when shown outside of running text

Courier bold
Used for:
- Specific user input (such as commands)

Courier italic
Used in procedures for:
- Variables on command line
- User input variables

<> Angle brackets enclose parameter or variable values supplied by the user

[] Square brackets enclose optional values

| Vertical bar indicates alternate selections - the bar means “or”

{} Braces indicate content that you must specify (that is, x or y or z)

... Ellipses indicate nonessential information omitted from the example

Where to get help
EMC support, product, and licensing information can be obtained as follows.
**Product information**—For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Powerlink website (registration required) at: —
http://support.EMC.com

**Technical support**—For technical support, go to EMC Online Support and choose Support. On the Support page, you will see several options, including one for making a service request. Note that to open a service request, you must have a valid support agreement. Contact your EMC sales representative for details about obtaining a valid support agreement or with questions about your account.

**The author of this TechBook**
This TechBook was written by Paul Pendle, an EMC employee working for the Symmetrix Mainframe Engineering team based at Hopkinton, Massachusetts. Paul Pendle has over 35 years of experience in databases, hardware, software and operating systems, both from a database administrator perspective and from a systems administrator/systems programming perspective.

**Your comments**
Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Send your opinions of this document to:

techpubcomments@emc.com
An Introduction to DB2 for z/OS

This chapter provides an overview of IBM’s DB2 database management system running on the IBM z/OS operating system.

- Overview .................................................................................................................. 20
- Using storage management for managing DB2 systems.................... 26
Overview

The z/OS processing environment provides a highly scalable and highly available processing platform. Parallel sysplex implementations provide high scalability and availability using DB2 data sharing capabilities. The combination of parallel sysplex and DB2 data sharing functions makes it a preferred choice for providing database server functions.

DB2 for z/OS has been available for over 20 years. It is the premier relational database implementation for mainframe environments. Throughout its lifespan, IBM has continually enhanced DB2 in each release to provide unparalleled scalability and availability features.

The architectural robustness of DB2, scalability, and availability functions have positioned it as a cornerstone of mainframe software architectures and application infrastructures, making it a critical component to many corporate application systems.

DB2 for z/OS can be configured in multiple ways. The requirement for 24/7 operations and the capacity of the host that contains the DB2 instance determines, in part, how the DB2 environment must be configured.

DB2 system elements

DB2 for z/OS consists of three basic and multiple optional components. These components exist as started tasks in the z/OS environment. Figure 1 on page 21 shows an example of how a DB2 system could be configured.
An Introduction to DB2 for z/OS

Figure 1  DB2 started tasks

In this example, DSN1MSTR, DSN1DBM1, and IRLMPROC are the basic system components that are required for DB2 to be operational.

- DSN1MSTR is responsible for handling system services. It coordinates the attachment of DB2 to other systems, such as CICS, IMS/DC, or TSO, and handles all logging activities: Physical logging, log archival, and BSDS.

- DSN1DBM1 manages database services. It processes SQL statements and manages data buffers. It contains the core logic of the database management system.

- IRLMPROC maintains data integrity by providing locking services. It handles locking and deadlock situations.

- DSN1WLM can start multiple address spaces, as required, to handle requests for Stored Procedures.

- DSN1DIST is the Distributed Data Facility (DDF). DB2 applications can use DDF to access data at other DB2 locations. DSN1DIST is used to create and manage distributed database applications.

- DSN1ADMT is the Admin Scheduler. Starting with DB2 V9, each DB2 system or data sharing member now has this started task. The Admin Scheduler uses a VSAM RRDS to contain a task list. All members of a data sharing group share the same task list dataset, however each member has a separate unique started task name.
DB2 data elements

DB2 maintains a set of data elements that are critical to the operation of the DB2 system. These data elements are the DB2 catalog, directory, bootstrap dataset (BDS), and logs. DB2 data elements for a non-data sharing DB2 environment are shown in Figure 2, which is followed by a description of each data element.

Figure 2  DB2 data elements: Non-data sharing

The DB2 catalog consists of tables of data describing everything defined to the DB2 instance, including table spaces, indexes, tables, copies of table spaces and indexes, storage groups, and so on. The system database, DSNDB06, contains the DB2 catalog.

The DB2 directory consists of a set of DB2 table structures in five table spaces in system database DSNDB01. Data relating to log ranges, plans, packages, databases, and utilities is contained within DSNDB01.

Active logs contain data changes, pre- and post-data images and other significant events. Active logs work in single or dual mode.

Archive logs are offloaded copies of the active logs and may be required for recovering DB2 objects with the LOGONLY option, as well as other utilities and DB2 rollback. Archive logs may be configured in single or dual mode.
The bootstrap dataset (BSDS) is a VSAM KSDS file that contains information vital to DB2. This dataset is the heart of the DB2 system. It serves as a repository for information critical to DB2 operation and recovery:

- An inventory of all active and archive log datasets known to DB2. This information is used to track the active and archive log datasets. DB2 also uses this information to locate log records to satisfy log read requests during normal DB2 system activity and during restart and recovery processing.
- An inventory of all recent DB2 checkpoint activity. DB2 uses this information during restart processing.
- The distributed data facility (DDF) communication record.
- The bootstrap dataset also includes a table of IP addresses that identify a host within the TCP/IP network.
- System point-in-time recovery information.

The work file database is used as storage for processing SQL statements that require working space, such as that required for a sort. In a non-data sharing environment, this database is DSNDB07. Each member of a data sharing group has a work file database with a unique name.

**DB2 data sharing**

A DB2 data sharing environment consists of multiple DB2 instances that share a single DB2 catalog, DB2 directory, and application table spaces that are contained within the data sharing environment. DB2 instances that are in the same data sharing group have access to all application data concurrently. Each member of the group must have its own set of local buffer pools, BSDS, active logs, archive logs, and work datasets.

Due to increasing demand for a 24/7 availability, data sharing has gained a lot of traction on z/OS parallel sysplex operating platforms. Since all members of a data sharing group can access all application data, one member of a group can assume the workload from another member that may be unavailable, due to maintenance or other planned or unplanned outages.

**Figure 3 on page 24** is a diagram of a DB2 data sharing group with two members. Note that each member of the group has its own local buffer pools, BSDS, active logs, and archive logs. Though not shown
here, each member also has its own set of started tasks, which are shown in Figure 1 on page 21. Members of a data sharing group communicate through the Sysplex Coupling Facility and share the DB2 catalog and directory.

**Figure 3** DB2 data sharing system and data elements

**DB2 attachment facilities**

Attachment facilities provide interfaces between DB2 and other environments. They are used to begin a DB2 session with applications on other platforms. Figure 4 on page 25 shows the z/OS attachment facilities that can interface with DB2.
The following describes the attachment facilities that communicate with DB2:

- TSO (Time Sharing Option)
- CICS
- IMS
- CAF
- RRS
Using storage management for managing DB2 systems

Standard DB2 backup, recovery, and cloning methods can be difficult to manage and time consuming. EMC provides alternative solutions to traditional DB2 utilities for cloning systems, backup and recovery, and disaster recovery or business continuity.

Cloning DB2 environments and objects

EMC® technology enables creation of an instant point-in-time copy of a DB2 system. The cloned copy is an identical environment to its source and can be used for other processing purposes, such as backup, recovery, offline reporting, and testing.

Object cloning is an alternative to cloning an entire DB2 system. Using EMC technology, it is possible to clone any number of DB2 objects within the same, or to a different, DB2 environment.

Backup and recovery

Backup and recovery operations using DB2 utilities require intervention by experienced personnel and can be labor-intensive in a large DB2 instance. Recovery of large DB2 instances, such as in an SAP or PeopleSoft environment, is especially complex because the entire system is a large referential set. All data in the set needs to be recovered together and to the same recovery point.

Dynamic manipulation of objects and application-maintained referential integrity further complicate recovery efforts. Traditional DB2 recovery techniques require multiple passes of the data, which can greatly impact recovery times. Such techniques are generally unworkable in large DB2 environments due to the time required to recover all objects. EMC hardware and software can be used to make the process faster and more effective.
This chapter introduces the EMC foundation products discussed in this document that are used in a DB2 for z/OS environment using Symmetrix arrays:

- Introduction ................................................................. 28
- Symmetrix hardware and EMC Enginuity features .............. 30
- ResourcePak Base for z/OS ............................................. 36
- SRDF family of products for z/OS .................................. 41
- EMC AutoSwap ............................................................. 53
- EMC TimeFinder family products for z/OS ......................... 56
- EMC z/OS Storage Manager .......................................... 61
- Symmetrix Management Console .................................. 62
- Symmetrix Performance Analyzer ................................... 63
- Unisphere for VMAX ..................................................... 64
- Virtual Provisioning ....................................................... 65
- Fully Automated Storage Tiering .................................... 65
- Data at Rest Encryption ................................................ 66
EMC provides many hardware and software products that support application environments on Symmetrix® systems. The following products, which are highlighted and discussed, were used and/or tested with the DB2 for z/OS. This chapter provides a technical overview of the EMC products used in this document.

**EMC Symmetrix**—EMC offers an extensive product line of high-end storage solutions targeted to meet the requirements of mission-critical databases and applications. The Symmetrix product line includes the DMX® Direct Matrix Architecture® and the VMAX® Virtual Matrix® Family. The Symmetrix controller is a fully redundant, high-availability storage processor, providing nondisruptive component replacements and code upgrades. The Symmetrix subsystem features high levels of performance, data integrity, reliability, and availability.

**EMC Enginuity™ Operating Environment**—Enginuity enables interoperation between current Symmetrix arrays and previous generations of Symmetrix arrays and enables them to connect to a large number of server types, operating systems and storage software products, and a broad selection of network connectivity elements and other devices, including Fibre Channel, ESCON, FICON, directors and switches.

**EMC Mainframe Enabler**—Mainframe Enabler is a package that contains all the Symmetrix API runtime libraries for all EMC MF software. These software packages can be used to monitor device configuration and status and to perform control operations on devices and data objects within a storage complex.

**EMC Symmetrix Remote Data Facility (SRDF®)**—SRDF is a business continuity software solution that replicates and maintains a mirror image of data at the storage block level in a remote Symmetrix system. The SRDF host component (HC) is a licensed feature product and when licensed in the SCF address space, it provides command sets to inquire on and manipulate remote Symmetrix relationships.

**EMC SRDF consistency groups**—An SRDF Consistency Group is a collection of related Symmetrix devices that are configured to act in unison to maintain data integrity. The devices in consistency groups can be spread across multiple Symmetrix systems.
EMC TimeFinder®—TimeFinder is a family of products that enables both volume-based replication and dataset replication within a single Symmetrix system. Data is copied from Symmetrix devices using array-based resources without using host CPU or I/O. The source Symmetrix devices remain online for regular I/O operations, while the copies are created. The TimeFinder family has three separate and distinct software products, TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap:

- TimeFinder/Mirror enables users to configure special devices, called business continuance volumes (BCVs), to create a mirror image of Symmetrix standard devices. Using BCVs, TimeFinder creates a point-in-time copy of data that can be repurposed. The TimeFinder/Mirror component extends the basic API command set of Mainframe Enabler to include commands that specifically manage Symmetrix BCVs and standard devices.

- TimeFinder/Clone or, more fully, TimeFinder/Clone Mainframe Snap Facility (TFCMSF) enables users to make copies of data from source volumes to target volumes without consuming mirror positions within the Symmetrix array. The data is available to a target’s host immediately upon activation, even if the copy process has not completed. Data may be copied from a single source device to as many as 16 target devices. A source device can be either a Symmetrix standard device or a BCV device.

- TimeFinder/Snap enables users to utilize special devices in the Symmetrix array called virtual devices (VDEVs) and save area devices (SAVDEVS). These devices can be used to make pointer-based, space-saving copies of data simultaneously on multiple target devices from a single source device. The data is available to a target’s host immediately upon activation. Data may be copied from a single source device to as many as 128 VDEVs. A source device can be either a Symmetrix standard device or a BCV device. A target device is a VDEV. A SAVDEV is a special device, without a host address, that is used to hold the changing contents of the source or target device after the snap is activated.

EMC Change Tracker—EMC Symmetrix Change Tracker software measures changes to data on a Symmetrix volume or group of volumes. Change Tracker software is often used as a planning tool in the analysis and design of configurations that use the EMC TimeFinder or SRDF components to store data at remote sites.
Symmetrix hardware and EMC Enginuity features

Symmetrix hardware architecture and the EMC Enginuity operating environment are the foundation for the Symmetrix storage platform. This environment consists of the following components:

- Symmetrix hardware
- Enginuity-based operating functions
- Mainframe Enabler
- Symmetrix application program interface (API) for mainframe
- Symmetrix-based applications
- Host-based Symmetrix applications
- Independent software vendor (ISV) applications

Figure 5 shows the relationship between these software layers and the Symmetrix hardware.

Symmetrix VMAX platform

The EMC Symmetrix VMAX Family arrays with Enginuity is the latest entry to the Symmetrix product line. Built on the strategy of simple, intelligent, modular storage, it incorporates a new scalable fabric interconnect design that allows the storage array to seamlessly
Symmetrix hardware and EMC Enginuity features grow from an entry-level configuration into the world’s largest storage system. Symmetrix VMAX arrays provide improved performance and scalability for demanding enterprise storage environments while maintaining support for EMC’s broad portfolio of platform software offerings.

Symmetrix VMAX arrays also maintain customer expectations for high-end storage in terms of availability. High-end availability is more than just redundancy. It means nondisruptive operations and upgrades, and being always online. Symmetrix VMAX systems provide:

- Nondisruptive expansion of capacity and performance at a lower price point
- Sophisticated migration for multiple storage tiers within the array
- The power to maintain service levels and functionality as consolidation grows
- Simplified control for provisioning in complex environments

Many of the new features provided by the EMC Symmetrix VMAX platform can reduce operational costs for customers deploying DB2 for z/OS solutions, as well as enhance functionality to enable greater benefits. This document details those features that provide significant benefits to customers utilizing DB2 for z/OS.

Figure 6 on page 32 illustrates the architecture and interconnection of the major components in the Symmetrix VMAX storage system.
EMC Enginuity operating environment

EMC Enginuity is the operating environment for all Symmetrix storage systems. Enginuity manages and ensures the optimal flow and integrity of data through the different hardware components. It also manages Symmetrix operations associated with monitoring and optimizing internal data flow. This ensures the fastest response to the user’s requests for information, along with protecting and replicating data. Enginuity provides the following services:

- Manages system resources to intelligently optimize performance across a wide range of I/O requirements.
- Ensures system availability through advanced fault monitoring, detection, and correction capabilities and provides concurrent maintenance and serviceability features.
- Offers the foundation for specific software features available through EMC disaster recovery, business continuity, and storage management software.
EMC Foundation Products

- Provides functional services for both Symmetrix-based functionality and for a large suite of EMC storage application software.
- Defines priority of each task, including basic system maintenance, I/O processing, and application processing.
- Provides uniform access through APIs for internal calls, and provides an external interface to allow integration with other software providers and ISVs.

Symmetrix features for mainframe

This section discusses supported Symmetrix features for mainframe environments.

I/O support features

Parallel Access Volume (PAV)—Parallel Access Volumes are implemented within a z/OS environment. They allow one I/O to occur for each base unit control block (UCB), and one for each statically or dynamically assigned alias UCB. These alias UCBs allow parallel I/O access for volumes. Current Enginuity releases provide support for static, dynamic, and hyperPAVs. HyperPAVs allow fewer aliases to be defined within a logical control unit. With hyperPAVs, aliases are applied to the base UCBs (devices) that require them the most at the time of need.

Multiple Allegiance (MA)—While PAVs facilitate multiple parallel accesses to the same device from a single LPAR, Multiple Allegiance (MA) allows multiple parallel nonconflicting accesses to the same device from multiple LPARs. Multiple Allegiance I/O executes concurrently with PAV I/O. The Symmetrix storage system treats them equally and guarantees data integrity by serializing writes where extent conflicts exist.

Host connectivity options—Mainframe host connectivity is supported through Fibre Channel, ESCON, and FICON channels. Symmetrix storage systems appear to mainframe operating systems as any of the following control units: IBM 3990, IBM 2105, and IBM 2107. The physical storage devices can appear to the mainframe operating system as any mixture of different sized 3380 and 3390 devices.
ESCON support—Enterprise Systems Connection (ESCON) is a fiber-optic connection technology that interconnects mainframe computers, workstations, and network-attached storage devices across a single channel, and supports half-duplex data transfers. ESCON may also be used for handling Symmetrix Remote Data Facility (SRDF) remote links.

FICON support—Fiber Connection (FICON) is a fiber-optic channel technology that extends the capabilities of its previous fiber optic channel standard, ESCON. Unlike ESCON, FICON supports full-duplex data transfers and enables greater throughput rates over longer distances. FICON uses a mapping layer based on technology developed for Fibre Channel and multiplexing technology, which allows small data transfers to be transmitted at the same time as larger ones. With Enginuity 5670 and later, Symmetrix storage systems support FICON ports. With the Enginuity service release 5874.207, VMAX arrays support 8 Gb FICON connectivity.

zHPF support—System z10 High Performance FICON (zHPF) represents the latest enhancement to the FICON interface architecture that aims at offering an improvement in the performance of online transaction processing (OLTP) workloads. Customers that are presently channel-constrained running heavy DB2 workloads using a 4K page size will reap the greatest benefit from this feature. zHPF is a chargeable, licensable feature with Enginuity service release 5874.207.

Fibre Channel support—Fibre Channel is a supported option in SRDF environments.

GigE support—GigE is a supported option in SRDF environments. Symmetrix GigE directors in an SRDF environment provide direct, end-to-end TCP/IP connectivity for remote replication solutions over extended distances with in-built compression. This removes the need for costly FC to IP converters and helps utilize the existing IP infrastructures without major disruptions.

**Data protection options**

Symmetrix storage systems incorporate many standard features that provide a higher level of data availability than conventional Direct Access Storage Devices (DASD). These options ensure a greater level of data recoverability and availability. They are configurable at the logical-volume level so different protection schemes can be applied to different classes of data within the same Symmetrix storage system.
on the same physical device. Customers choose data protection options, such as the following, to match their availability requirements:

- Mirroring RAID 1 or RAID 10
- RAID 6 (6+2) and RAID 6 (14+2)
- RAID 5 (3+1) and RAID 5 (7+1)
- Symmetrix Remote Data Facility (SRDF)
- TimeFinder
- Dynamic Sparing
- Global Sparing

**Other features**

Other IBM-supported compatibility features include:

- Channel Command Emulation for IBM ESS 2105/2107
- Concurrent Copy
- Peer to Peer Remote Copy (PPRC)
- PPRC/XRC Incremental Resync
- Extended Remote Copy (XRC) with multireader
- Dynamic Channel Path Management (DCM)
- Dynamic Path Reconnection (DPR) support
- Host Data compression
- Logical Path and Control Unit Address support (CUADD)
- Multi-System imaging
- Partitioned dataset (PDS) Search Assist
- FlashCopy Version 1 and 2
- High Performance Ficon (zHPF) and multitrack zHPF
- Extended Address Volumes (EAV)
ResourcePak Base for z/OS

EMC ResourcePak® Base for z/OS is a software facility that enables communication between mainframe-based applications (provided by EMC or ISVs) and a Symmetrix storage system. ResourcePak Base is designed to improve performance and ease of use of mainframe-based Symmetrix applications.

ResourcePak Base delivers EMC Symmetrix Control Facility (EMCSCF) for IBM and IBM-compatible mainframes. EMCSCF provides a uniform interface for EMC and ISV software products, where all products are using the same interface at the same function level. EMCSCF delivers a persistent address space on the host that facilitates communication between the host and the Symmetrix array, as well as other EMC-delivered, and partner-delivered applications.

Figure 7 logically depicts the relationships between the SCF address space and the other software components accessing the Symmetrix subsystem.

Figure 7  z/OS SymmAPI architecture

ResourcePak Base is the delivery mechanism for the EMC Symmetrix Applications Programming Interface for z/OS (SymmAPI™-MF). ResourcePak Base provides a central point of control by giving software a persistent address space on the mainframe for SymmAPI-MF functions that perform tasks such as the following:
EMC Foundation Products

- Maintaining an active repository of information about EMC Symmetrix devices attached to z/OS environments and making that information available to other EMC products.
- Performing automation functions.
- Handling inter-LPAR (logical partition) communication through the Symmetrix storage system.

ResourcePak Base provides faster delivery of new Symmetrix functions by EMC and ISV partners, along with easier upgrades. It also provides the ability to gather more meaningful data when using tools such as TimeFinder/Mirror QUERY because device status information is now cached along with other important information.

ResourcePak Base for z/OS is a prerequisite for EMC mainframe applications, like the TimeFinder/Clone Mainframe SNAP Facility or SRDF Host Component for z/OS, and is included with these products.

With EMC Mainframe Enabler software V7.0, ResourcePak Base, TimeFinder, and SRDF are shipped in a single distribution.

Features

ResourcePak Base provides the following functionality with EMCSCF:
- Cross-system communication
- Nondisruptive SymmAPI-MF refreshes
- Save Device Monitor
- SRDF/A Monitor
- Group Name Service (GNS) support
- Pool management
- SRDF/AR resiliency
- SRDF/A multisession consistency
- SWAP services
- Recovery services
- FlashCopy emulation (Enginuity 5772 or earlier)
- Licensed feature code management
Cross-system communication

Inter-LPAR communication is handled by the EMCSCF cross-system communication (CSC) component. CSC uses a Symmetrix storage system to facilitate communications between LPARs. Several EMC Symmetrix mainframe applications use CSC to handle inter-LPAR communications.

Nondisruptive SymmAPI-MF refreshes

EMCSCF allows the SymmAPI-MF to be refreshed nondisruptively. Refreshing SymmAPI-MF does not impact currently executing applications that use SymmAPI-MF, for example, SRDF Host Component for z/OS or TimeFinder.

Save Device Monitor

The Save Device Monitor periodically examines the consumed capacity of the device pool (SNAPPOOL) used by TimeFinder/Snap with the VDEV licensed feature code enabled. The Save Device Monitor also checks the capacity of the delta set extension pool (DSEPOOL) used by SRDF/A.

The Save Device Monitor function of EMCSCF provides a way to:
- Automatically check space consumption thresholds.
- Trigger an automated response that is tailored to the specific needs of the installation.

SRDF/A Monitor

The SRDF/A Monitor in ResourcePak Base is designed to:
- Find EMC Symmetrix controllers that are running SRDF/A.
- Collect and write SMF data about those controllers.

After ResourcePak Base is installed, the SRDF/A Monitor is started as a subtask of EMCSCF.

Group Name Service support

ResourcePak Base includes support for Symmetrix Group Name Service (GNS). Using GNS, you can define a device group once and then use that single definition across multiple EMC products on multiple platforms. This means that you can use a device group defined through GNS with both mainframe and open systems-based EMC applications. GNS also allows you to define group names for volumes that can then be operated upon by various other commands.
Pool management

With ResourcePak Base 5.7 or later, generalized device pool management is a provided service. Pool devices are a predefined set of devices that provide a pool of physical space. Pool devices are not host-accessible. The CONFIGPOOL commands allow management of SNAPPOOLs or DSEPOOLs with CONFIGPOOL batch statements.

SRDF/AR resiliency

SRDF/AR can recover from internal failures without manual intervention. Device replacement pools for SRDF/AR (or SARPOOLs) are provided to prevent SRDF/AR from halting due to device failure. In effect, SARPOOLs are simply a group of devices that are unused until SRDF/AR needs one of them.

SRDF/A Multi-Session Consistency

SRDF/A Multi-Session Consistency (MSC) is a task in EMCSCF that ensures remote R2 consistency across multiple Symmetrix storage systems running SRDF/A. MSC provides the following:

- Coordination of SRDF/A cycle switches across systems.
- Up to 24 SRDF groups in a multi-session group.
- One SRDF/A session and one SRDF/A group per Symmetrix storage system when using Enginuity 5X70.
- With Enginuity 5x71 and later, SRDF/A groups are dynamic and are not limited to one per Symmetrix storage system. Group commands of ENABLE, DISPLAY, DISABLE, REFRESH, and RESTART are now available.

SWAP services

ResourcePak Base deploys a SWAP service in EMCSCF. It is used by EMC AutoSwap™ for planned outages with the ConGroup Continuous Availability Extensions (CAX).

Recovery services

Recovery service commands allow you to perform recovery on local or remote devices (if the links are available for the remote devices).

FlashCopy support in SCF

FlashCopy version 1 and version 2 support is enabled in EMCSCF through a LFC when using Enginuity 5772 or earlier. FlashCopy support is enabled in Enginuity 5773 and later and does not require a license key in EMCSCF.
EMCSCF manages licensed feature codes (LFCs) to enable separately chargeable features in EMC software. These features require an LFC to be provided during the installation and customization of EMCSCF. LFCs are available for:

- Symmetrix Priority Control
- Dynamic Cache Partitioning
- AutoSwap (ConGroup with AutoSwap Extensions)—separate LFCs are required for planned and unplanned swaps
- EMC Compatible Flash (Host Software Emulation)
- EMC z/OS Storage Manager (EzSM)
- SRDF/Asynchronous (MSC)
- SRDF/Automated Replication
- SRDF/Star
- TimeFinder/Clone Mainframe Snap Facility
- TimeFinder/Consistency Group
- TimeFinder/Snap (VDEV)
SRDF family of products for z/OS

At the conceptual level, SRDF is mirroring (RAID level 1) one logical disk device (the primary source/R1 within a primary Symmetrix storage system) to a second logical device (the secondary target/R2, in a physically separate secondary Symmetrix storage system) over ESCON, Fibre Channel, or GigE high-speed communication links. The distance separating the two Symmetrix storage systems can vary from a few feet to thousands of miles. SRDF is the first software product for the Symmetrix storage system. Its basic premise is that a remote mirror of data (data in a different Symmetrix storage system) can serve as a valuable resource for:

- Protecting data using geographical separation.
- Giving applications a second location from which to retrieve data should the primary location become unavailable for any reason.
- Providing a means to establish a set of volumes on which to conduct parallel operations, such as testing or modeling.

SRDF has evolved to provide different operation modes (synchronous, adaptive copy—write pending mode, adaptive copy—disk mode, and asynchronous mode). More advanced solutions have been built upon it, such as SRDF/Automated Replication and SRDF/Star, Cascaded SRDF, and SRDF/EDP.

Persistent throughout these evolutionary stages has been control of the SRDF family of products by the mainframe-based application called SRDF Host Component. SRDF Host Component is a control mechanism through which all SRDF functionality is made available to the mainframe user. EMC Consistency Group for z/OS is another useful feature for managing dependent-write consistency across Symmetrix links with one or more mainframes attached.
Figure 8 indicates that the modules on the right plug into one of the modules in the center as an add-on function. For example, SRDF Consistency Group is a natural addition to customers running SRDF in synchronous mode.

**SRDF Host Component for z/OS**

SRDF Host Component for z/OS, along with ResourcePak Base for z/OS (API services module), is delivered when ordering a member of the SRDF product family. For more information about SRDF technology in general please visit:

https://support.emc.com/products/1848

**SRDF mainframe features**

SRDF mainframe features include the following:

- Ability to deploy SRDF solutions across the enterprise: SRDF Host Component can manage data mirroring for both CKD and FBA format disks. In these deployments, both a mainframe and one or more open system hosts are attached to the primary side of
the SRDF relationship. Enterprise SRDF deployments can be controlled either by mainframe hosts or by open systems hosts, though the toolsets are different in each environment.

- Support for either ESCON or FICON host channels regardless of the SRDF link protocol employed: SRDF is a protocol that links Symmetrix systems, mirroring data on both sides of the communications link. Host connectivity to the Symmetrix storage system (ESCON or FICON) is independent of the protocols used in moving data using the SRDF links. SRDF supports all the standard link protocols: ESCON, Extended ESCON, Fibre Channel, and GigE.

- Software support for taking an SRDF link offline: SRDF Host Component has a software command that can take an SRDF link offline independent of whether it is taking the target volume offline. This feature is useful if there are multiple links in the configuration and only one is experiencing issues, for example too many bounces (sporadic link loss) or error conditions. In this case, it is unnecessary to take all links offline, when taking the one in question offline is sufficient.

- SRDF Host Component additional interfaces: Besides the console interface, all features of SRDF Host Component can be employed using REXX scripting and/or the Stored Procedure Executive (SPE), a powerful tool for automating repeated processes.

Concurrent SRDF and SRDF/Star

SRDF/Star is built upon several key technologies:

- Dynamic SRDF
- Concurrent SRDF
- ResourcePak Base for z/OS
- SRDF/Synchronous
- SRDF/Asynchronous
- Consistency Group
- Certain features within Enginuity

SRDF/Star provides advanced multisite business continuity protection that augments Concurrent SRDF/tables and SRDF/A operations from the same primary volumes with the ability to
incrementally establish an SRDF/A session between the two remote sites in the event of a primary site outage. This capability is only available through SRDF/Star software.

SRDF/Star is a combination of mainframe host software and Enginuity functionality that operates concurrently. Figure 9 depicts a classic three-site SRDF/Star configuration.

The concurrent configuration option of SRDF/A provides the ability to restart an environment at long distances with minimal data loss, while simultaneously providing a zero data loss restart capability at a local site. Such a configuration provides protection for both a site disaster and a regional disaster, while minimizing performance impact and loss of data.

In a Concurrent SRDF/A configuration without SRDF/Star functionality, the loss of the primary A site would normally mean that the long-distance replication would stop, and data would no longer propagate to the C site. Data at site C would continue to age as production was resumed at site B. Resuming SRDF/A between sites B and C would require a full resynchronization to re-enable disaster recovery protection. This consumes both time and resources and results in a prolonged period of time without having normal DR protection.
SRDF/Star provides a rapid re-establishment of cross-site protection in the event of a primary site (A) failure. Rather than a full resynchronization between sites B and C, SRDF/Star provides a differential B to C synchronization, dramatically reducing the time to remotely protect the new production site. SRDF/Star also provides a mechanism for the user to determine which site (B or C) has the most current data in the event of a rolling disaster affecting site A. In all cases, the choice of which site to use in the event of a failure is left to the customer’s discretion.

Multi-Session Consistency

In SRDF/A environments, consistency across multiple Symmetrix systems for SRDF/A sessions is provided by the Multi-Session Consistency (MSC) task that executes in the EMCSCF address space. MSC provides consistency across as many as 24 SRDF/A sessions and is enabled by a Licensed Feature Code.

SRDF/AR

SRDF/Automated Replication (SRDF/AR) is an automated solution that uses both SRDF and TimeFinder to provide periodic, asynchronous replication of a restartable data image. In a single-hop SRDF/AR configuration, the magnitude of controlled data loss depends on the cycle time chosen. However, if greater protection is required, a multi-hop SRDF/AR configuration can provide long-distance disaster restart with zero data loss using a middle or bunker site.

EMC Geographically Dispersed Disaster Restart

EMC Geographically Dispersed Disaster Restart (GDDR) is a mainframe software product that automates business recovery following both planned outages and disasters, including the total loss of a data center. EMC GDDR achieves this goal by providing monitoring, automation, and quality controls for many EMC and third-party hardware and software products required for business restart.
Because EMC GDDR restarts production systems following disasters, it does not reside on the same LPAR that it protects. EMC GDDR resides on separate logical partition (LPAR) from the host system that is running application workloads.

In a three-site SRDF/Star with AutoSwap configuration, EMC GDDR is installed on a control LPAR at each site. Each EMC GDDR node is aware of the other two EMC GDDR nodes through network connections between each site. This awareness allows EMC GDDR to:

- Detect disasters
- Identify survivors
- Nominate the leader
- Recover business at one of the surviving sites

To achieve the task of business restart, EMC GDDR automation extends well beyond the disk level (on which EMC has traditionally focused) and into the host operating system. It is at this level that sufficient controls and access to third-party software and hardware products exist, enabling EMC to provide automated recovery services.

EMC GDDR’s main activities include:

- Managing planned site swaps (workload and DASD) between the primary and secondary sites and recovering the SRDF/Star with AutoSwap environment.
- Managing planned site swaps (DASD only) between the primary and secondary sites and recovering the SRDF/Star with AutoSwap environment.
- Managing the recovery of the SRDF environment and restarting SRDF/A in the event of an unplanned site swap.
- Active monitoring of the managed environment and responding to exception conditions.

---

**SRDF Enterprise Consistency Group for z/OS**

An SRDF consistency group is a collection of devices logically grouped together to provide consistency. Its purpose is to maintain data integrity for applications that are remotely mirrored, particularly those that span multiple RA groups or multiple Symmetrix storage systems. The protected applications may be comprised of multiple heterogeneous data resource managers spread
across multiple host operating systems. It is possible to span mainframe LPARs, UNIX, and Windows servers. These heterogeneous platforms are referred to as hosts.

If a primary volume in the consistency group cannot propagate data to its corresponding secondary volume, EMC software suspends data propagation from all primary volumes in the consistency group. The suspension halts all data flow to the secondary volumes and ensures a dependent-write consistent secondary volume copy at the point in time that the consistency group tripped.

The dependent-write principle concerns the logical dependency between writes that are embedded by the logic of an application, operating system, or database management system (DBMS). This notion is that a write will not be issued by an application until a prior, related write has completed (a logical dependency not a time dependency). Some aspects of this notion are:

- Inherent in all DBMS
- DB2 page (table space) write is a dependent write based on a successful log write
- Applications can also use this technology
- Power failures create dependent-write consistent images

DBMS restart transforms a dependent-write consistent data state to a transactionally consistent data state.

When the amount of data for an application becomes very large, the time and resources required for host-based software to protect, back up, or execute decision-support queries on these databases becomes critical. In addition, the time required to shut down those applications for offline backup is no longer acceptable, and alternative implementations are required. One alternative is SRDF Consistency Group technology which allows users to remotely mirror the largest data environments and automatically create dependent-write consistent, restartable copies of applications in seconds without interruption to online services.

Disaster restart solutions that use consistency groups provide remote restart with short recovery time objectives. SRDF synchronous configurations provide zero data loss solutions using consistency groups. Zero data loss implies that all completed transactions at the beginning of a disaster will be available at the target storage system after restart.
An SRDF Consistency Group has two methodologies to preserve a dependent-write consistent image while providing a synchronous disaster restart solution with a zero data loss scenario. These two methodologies are described in the following two sections.

**ConGroup using IOS and PowerPath**

This methodology preserves a dependent-write consistent image using IOS on mainframe hosts and PowerPath® on open systems hosts. This method requires all hosts have connectivity to all involved Symmetrix storage systems, either through direct connections or indirectly through one or more SAN configurations. These hosts are not required to have the logical devices visible. A path (gatekeeper) to each involved Symmetrix storage system is sufficient. The consistency group definition, software, and licenses must reside on all hosts involved in the consistency group. The read and write I/Os are both held with the IOS and PowerPath methodology.

**ConGroup using SRDF/ECA**

This is the preferred methodology with ConGroup. It preserves a dependent-write consistent image using SRDF Enginuity Consistency Assist (SRDF-ECA). This method requires a minimum of one host having connectivity to all involved Symmetrix storage systems, either through direct connections or indirectly through one or more storage network configurations. EMC recommends having at least two such hosts for redundancy purposes. In the event of a host failure, the second host can automatically take over control of the consistency functions. These hosts are referred to as control hosts, and are the only hosts required to have the consistency group definition, software, and licenses. During a ConGroup trip, SRDF-ECA defers writes to all involved logical volumes in the consistency group. Subsequent read I/Os are held per logical volume once the first write on that volume is deferred. This is done only for a short period of time while the consistency group suspends transfer operations to the secondary volumes.

EMC recommends that SRDF-ECA mode be configured when using a consistency group in a mixed mainframe and open systems environment with both CKD and FBA (fixed block architecture) devices.

Tripping a consistency group can occur either automatically or manually. Scenarios in which an automatic trip would occur include:

- One or more primary volumes cannot propagate writes to their corresponding secondary volumes.
◆ The remote device fails.
◆ The SRDF directors on either the primary or secondary Symmetrix storage systems fail.

In an automatic trip, the Symmetrix storage system completes the write to the primary volume, but indicates that the write did not propagate to the secondary volume. EMC software, combined with Symmetrix Enginuity, intercepts the I/O and instructs the Symmetrix storage system to suspend all primary volumes in the consistency group from propagating any further writes to the secondary volumes. Once the suspension is complete, writes to all primary volumes in the consistency group continue normally, but are not propagated to the target side until normal SRDF mirroring resumes.

An explicit trip occurs when a `susp-cgrp` (suspend ConGroup) command is invoked using SRDF Host Component software. Suspending the consistency group creates an on-demand, restartable copy of the database at the secondary site. BCV devices synchronized with the secondary volumes are then split after the consistency group is tripped, creating a second dependent-write consistent copy of the data. During the explicit trip, SRDF Host Component issues the command to create the dependent-write consistent copy, but may require assistance from either IOSLEVEL or SRDF-ECA by means of the ConGroup software if I/O is received on one or more of the primary volumes, or if the Symmetrix commands issued are abnormally terminated before the explicit trip.

An SRDF Consistency Group maintains consistency within applications spread across multiple Symmetrix storage systems in an SRDF configuration by monitoring data propagation from the primary volumes in a consistency group to their corresponding secondary volumes. Consistency groups provide data integrity protection during a rolling disaster. The loss of an SRDF communication link is an example of an event that could be a part of a rolling disaster.

Figure 10 on page 50 depicts a dependent-write I/O sequence where a predecessor log write happens before a DB2 page flush from a database buffer pool. The log device and data device are on different Symmetrix storage systems with different replication paths. Figure 10 also demonstrates how rolling disasters are prevented using SRDF Consistency Group technology.
1. A consistency group is defined containing volumes X, Y, and Z on the source Symmetrix array. This consistency group definition must contain all of the devices that need to maintain dependent-write consistency and reside on all participating hosts involved in issuing I/O to these devices. A mix of CKD (mainframe) and FBA (UNIX/Windows) devices can be logically grouped together. In many cases, the entire processing environment may be defined in a single consistency group to ensure dependent-write consistency.

2. The rolling disaster described previously begins.

3. The predecessor log write occurs to volume Z, but cannot be replicated to the remote site.

4. Since the predecessor log write to volume Z cannot be propagated to the remote Symmetrix system, a consistency group trip occurs.
   a. The source Symmetrix Enginuity captures the write I/O that initiated the trip event and defers all write I/Os to all logical devices within the consistency group on this Symmetrix system. The control host software is constantly polling all involved Symmetrix systems for such a condition.

---

**Figure 10** SRDF Consistency Group using SRDF-ECA

---

**Host 1**
- Consistency group
- Host component
- Symmetrix control Facility
- DBMS
- RDF-ECA

**Host 2**
- Consistency group
- Host component
- Symmetrix control Facility
- DBMS
- RDF-ECA

---

|x = DBMS data | y = Application data | z = Logs|

---

ICO-IMG-000106
b. Once a trip event is detected by the host software, an instruction is sent to all involved Symmetrix systems in the consistency group definition to defer all write I/Os for all logical devices in the group. This trip is not an atomic event. However, the process guarantees dependent-write consistency because of the integrity of the dependent-write I/O principle. Once a write I/O could not be received as complete to the host (the predecessor log write), the DBMS prevents the dependent I/O from being issued.

5. Once all of the involved Symmetrix storage systems have deferred the writes for all involved logical volumes of the consistency group, the host software issues a suspend action on the primary/secondary relationships for the logically grouped volumes, which immediately disables all replication of those grouped volumes to the remote site. Other volumes outside of the group are allowed to continue replicating, provided the communication links are available.

6. After the relationships are suspended, the completion of the predecessor write is acknowledged back to the issuing host. Furthermore, all I/Os that were held during the consistency group trip operation are released.

7. The dependent data write is issued by the DBMS and arrives at X but is not replicated to its secondary volume.

When a complete failure occurs from this rolling disaster, the dependent-write consistency at the secondary site is preserved. If a complete disaster does not occur and the failed links are reactivated, consistency group replication can be resumed. EMC recommends creating a copy of the dependent-write consistent image while the resume takes place. After the SRDF process reaches synchronization, the dependent-write consistent copy is achieved at the remote site.

**DB2 considerations for ConGroup**

Volumes in the ConGroup device list for the DB2 group must include all volumes where DB2 system data and all other application table spaces reside. DB2 system datasets that must be included are:

- BSDS01: Copy 1 of the bootstrap dataset.
- BSDS02: Copy 2 of the bootstrap dataset.
- LOGCOPY1.DS01: LOGCOPY1.DSnn: Copy 1 of the active logs and all datasets associated with this copy.
- LOGCOPY2.DS01: LOGCOPY2.DSn: Copy 2 of the active logs and all dataset associated with this copy.
- DSNDB01: DB2 directory datasets.
- DSNDB04: DB2 default database datasets.
- DSNDB06: DB2 catalog datasets.
- All user and vendor table spaces.
- Archives if long running UOW without commits are a concern.
- ICF catalogs for system and user data.
- DB2 libraries.

Figure 11 is a pictorial representation of a DB2 ConGroup configuration.

Restart in the event of a disaster or nondisaster

Two major circumstances require restartability or continuity of business processes as facilitated by SRDF: A true, unexpected disaster and an abnormal termination of processes on which data flow
depends. Both circumstances require that a customer immediately deploy the proper resource and procedures to correct the situation. It is generally the case that an actual disaster is more demanding of all necessary resources in order to successfully recover/restart.

**Disaster**

In the event of a disaster, where the primary Symmetrix storage system is lost, it is necessary to run database and application services from the DR site. This requires a host at the DR site. The first action is to write-enable the secondary devices (R2s). At this point, the host can issue the necessary commands to access the disks.

After the data is available to the remote host, the DB2 system may be restarted. DB2 performs an implicit recovery when restarted by resolving in-flight transactions. Transactions that were committed, but not completed, are rolled forward and completed using the information in the active logs. Transactions that have updates applied to the database, but were not committed, are rolled back. The result is a transactionally consistent database.

**Abnormal termination (not a disaster)**

An SRDF session can be interrupted by any situation that prevents the flow of data from the primary site to the secondary site (for example, a software failure, network failure, or hardware failure).

**EMC AutoSwap**

EMC AutoSwap provides the ability to move (swap) workloads transparently from volumes in one set of Symmetrix storage systems to volumes in other Symmetrix storage systems without operational interruption. Swaps may be initiated either manually as planned events or automatically as unplanned events (upon failure detection):

- Planned swaps facilitate operations such as nondisruptive building maintenance, power reconfiguration, DASD relocation, and channel path connectivity reorganization.

- Unplanned swaps protect systems against outages in a number of scenarios. Examples include: Power supply failures, building infrastructure faults, air conditioning problems, loss of channel connectivity, entire DASD system failures, operator error, or the consequences of intended or unintended fire suppression system discharge.
AutoSwap, with SRDF and SRDF Consistency Group, dramatically increase data availability.

In Figure 12, swaps are concurrently performed while application workloads continue in conjunction with EMC Consistency Group. This option protects data against unforeseen events, and ensures that swaps are unique, atomic operations that maintain dependent-write consistency.

**Figure 12**  AutoSwap before and after states

**AutoSwap highlights**

AutoSwap includes the following features and benefits:

- Testing on devices in swap groups to ensure validity of address-switching conditions. This supports grouping devices into swap groups and treats each swap group as a single-swap entity.

- Consistent swapping—Writes to the group are held during swap processing, ensuring dependent-write consistency to protect data and ensure restartability.

- Swap coordination across multiple z/OS images in a shared DASD or parallel sysplex environment. During the time when devices in swap groups are frozen and I/O is queued, AutoSwap reconfigures SRDF pairs to allow application I/O streams to be serviced by secondary SRDF devices. As the contents of UCBs are
swapped, I/O redirection takes place transparently to the applications. This redirection persists until the next Initial Program Load (IPL) event.

Use cases

AutoSwap can:

- Perform dynamic workload reconfiguration without application downtime.
- Concurrently swap large numbers of devices.
- Handle device group operations.
- Relocate logical volumes.
- Perform consistent swaps.
- Implement planned outages of individual devices or entire systems.
- React appropriately to unforeseen disasters if an unplanned event occurs.
- Protect against the loss of all DASD channel paths or an entire storage system. This augments the data integrity protection provided by Consistency Groups by providing continuous availability in the event of a failure affecting the connectivity to a primary device.
EMC TimeFinder family products for z/OS

For years, the TimeFinder family of products have provided important capabilities for the mainframe environment. TimeFinder has undergone changes over time, but the key features and functions are still the same as depicted by Figure 13.

![Figure 13 TimeFinder family of products for z/OS](ICO-IMG-000108)

**TimeFinder/Clone for z/OS**

TimeFinder/Clone for z/OS is documented as a component of the TimeFinder/Clone Mainframe SNAP Facility. It is the code and documentation associated with making full-volume snaps and dataset-level snaps. As such, they are space-equivalent copies. TimeFinder/Clone does not consume a mirror position, nor does it require the BCV flag for targets on the Symmetrix storage system. Certain TimeFinder/Mirror commands, such as Protected BCV Establish, are unavailable in the TimeFinder/Clone Mainframe Snap Facility. This command is one of pointer-based copy technology rather than mirror technology. Other protection mechanisms, such as RAID 5, are available for the target storage volumes as well.

Additional mainframe-specific capabilities of TimeFinder/Clone for z/OS include:

- Dataset-level snap operations.
EMC TimeFinder family products for z/OS

- Differential snap operations—These require only the changed data to be copied on subsequent full-volume snaps.
- Support for CONSISTENT SNAP operations—These make the target volumes dependent-write consistent and require the TimeFinder/Consistency Group product.
- Up to 16 simultaneous point-in-time copies of a single primary volume.
- Compatibility with STK Snapshot Copy and IBM Snap products, including reuse of its SIBBATCH syntax.
- TimeFinder Utility for z/OS—This utility conditions the ICF catalog by relabeling and recataloging entries to avoid issues associated with duplicate volume names in the mainframe environment. This utility is also delivered with TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap products.
- Compatibility with mainframe security mechanisms such as RACF.
- Integration with many mainframe-specific ISVs and their respective products.

TimeFinder/Snap for z/OS

TimeFinder/Snap for z/OS uses the code and documentation from TimeFinder/Clone, but with an important difference: Snaps made with this product are virtual snaps, meaning they take only a portion of the space a full-volume snap would. Invocation of this feature is through the keyword VDEV (Virtual Device). If the VDEV argument is used, only the pre-update image of the change data plus a pointer is kept on the target. This technique considerably reduces disk space usage on the target. This feature also provides for one or more named SNAPPOOLS that can be managed independently.

Mainframe-specific features of the TimeFinder/Snap for z/OS product include:
- The same code and syntax as the TimeFinder/Clone Mainframe SNAP Facility (plus the addition of the VDEV argument).
- The same features and functions as TimeFinder/Clone Mainframe SNAP Facility, and, therefore the same benefits.
- Full-volume support only (no dataset support).
EMC Foundation Products

- Up to 128 simultaneous point-in-time snaps of a single primary volume.
- The ability to create a point-in-time copy of a previously activated virtual device through a feature called Duplicate Snap. This allows the creation of multiple copies of a point-in-time copy. This feature requires Mainframe Enablers V7.2 software or later and is manageable using Symmetrix Management Console or Unisphere.
- ICF catalog conditioning with TimeFinder Utility for z/OS—This allows relabeling and recataloging entries and avoids issues associated with duplicate volume names in the mainframe environment.
- Compatibility with mainframe security mechanisms such as RACF.
- Integration with many mainframe-specific ISVs and their respective products.

---

**TimeFinder/Mirror for z/OS**

TimeFinder/Mirror for z/OS provides BCVs and the means by which a mainframe application can manipulate them. BCVs are specially tagged logical volumes manipulated by using these TimeFinder/Mirror commands: ESTABLISH, SPLIT, RE-ESTABLISH, and RESTORE.

Mainframe-specific features of the TimeFinder/Mirror product include:

- The TimeFinder Utility for z/OS, which conditions the VTOC, VTOCIX, the VVDS and the ICF catalog by re-labeling and re-cataloging entries, thereby avoiding issues associated with duplicate volume names and dataset names.
- The ability to create dependent-write consistent BCVs locally or remotely (with the plug-in module called TimeFinder/Consistency Group) without the need to quiesce production jobs.
- BCV operations important to IT departments include:
  - Using the BCV as the source for backup operations.
• Using the BCV for test LPARs with real data. The speed with which a BCV can be rebuilt means that multiple test cycles can occur rapidly and sequentially. Applications can be staged using BCVs before committing them to the next application refresh cycle.

• Using the BCV as the source for data warehousing applications rather than the production volumes. Because the BCVs are a point-in-time mirror image of the production data, they can be used as golden copies of data to be written and rewritten repeatedly.

◆ The use of SRDF/Automated Replication.

◆ The support for mainframe TimeFinder queries including the use of wildcard matching.

◆ The compatibility with mainframe security mechanisms such as RACF.

◆ The integration of DBMS utilities available from ISVs and their products.

◆ The integration of many mainframe-specific ISVs and their products.

On the Symmetrix VMAX platform, all TimeFinder/Mirror operations are emulated in Enginuity and converted to TimeFinder/Clone operations transparently.

**TimeFinder/CG**

TimeFinder/CG (Consistency Group) is a plug-in module for the TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap products. TimeFinder/CG provides consistency support for various TimeFinder family commands. TimeFinder/CG is licensed separately and uses a Licensed Feature Code implementation model.

TimeFinder/CG allows the use of the `CONSISTENT(YES)` parameter on TimeFinder/Clone and TimeFinder/Snap, as well as the `CONS` parameter on TimeFinder/Mirror `SPLIT` statements. This allows TimeFinder to create an instantaneous point-in-time copy of all the volumes being copied. The copy thus created is a dependent-write consistent copy which is in a state very similar to that which is created during a power outage. If a DB2 system is copied this way, a restartable image is created.
Consistent dataset snap

The consistent dataset snap capability for z/OS, offered through Enginuity 5875, is unique to EMC. Consistent dataset snap allows the user to obtain a dependent-write consistent image of multiple DB2 datasets using the TimeFinder/Clone MF Snap Facility SNAP DATASET function.

Currently, inter-dataset consistency is only achievable by requesting an exclusive enqueue on the source datasets, which is very impractical in production DB2 environments. The larger the production environment is, the more impractical it becomes. In addition, this approach does not provide for cross-sysplex consistency, another consideration in very large environments.

Other factors prohibiting this function in the past were that the SNAP dataset approach operated on a single dataset at a time, and the Enginuity extent snap took far too long to be executed under an Enginuity Consistency Assist (ECA) window. A further complication was that the extent snap did not allow for the separation of the establish and activate phases of the snap of a dataset (but was allowed for a full volume). This was critically needed and has now been provided.

Thus, the consistent extent snap feature, introduced with Enginuity 5875, provides the ability to do separate establish and activate extent-level snap operations, which in turn results in dataset snap processing on an entire group of datasets such that dependent-write consistency is ensured across the resulting group of target datasets.

A separate ACTIVATE statement can now follow SNAP dataset statements and CONSISTENT(YES) is now allowed on ACTIVATE as documented in detail in the EMC TimeFinder/Clone Mainframe SNAP Facility Product Guide.

TimeFinder/CG invokes ECA (Enginuity Consistency Assist) to hold I/Os while the copy is taken. There is little or no effect on the host application or database during this time.
The EMC z/OS Storage Manager (EzSM) is a mainframe software product providing storage management in a Symmetrix environment. EzSM provides mainframe storage managers and operations staff a flexible, z/OS-centric view of storage that presents both Symmetrix system-specific information and z/OS storage management data in a single easy-to-use 3270 interface.

With EzSM, users can discover and monitor the volumes in a Symmetrix subsystem, set alerts for volumes, summarize Symmetrix configuration information, and much more. EzSM is installed with SMP/E (System Modification Program Extended), an element of z/OS that is used to install most software products. EzSM logs user activity and records changes in SMP. Standard security packages and z/OS messaging are used, which allows customer automation packages to filter EzSM messages. Figure 14 is a logical view of EzSM functionality.

Figure 14  EMC z/OS Storage Manager functionality
Symmetrix Management Console

Symmetrix Management Console (SMC) employs a simple and intuitive Web-based user interface to administer the most common daily storage management functions for the Symmetrix array. The intention is that SMC can be used quickly and efficiently by operators of all experience levels.

Members of the mainframe community typically participate in a structured change control process to manage their storage environment with certainty and stability. When using the SMC, mainframe storage administrators can avoid consultation with EMC personnel on array change control activities and perform the actions themselves, removing one level of complexity in the change control process. It is anticipated that changes can be enacted in a more timely fashion and communication errors avoided when SMC is used by authorized customer administrators to directly perform array modifications.

SMC puts control of the following array activities into the hands of the mainframe storage administrator:

- Device creation and removal
- Device base and alias addressing
- Local and remote replication
- Quality of service
- Replication and quality of service monitoring
- Management of FAST™ policies and configurations

SMC is designed to deliver Symmetrix array management that is responsive to user controls and modest on server resource requirements. As a consequence of this design mandate, one SMC instance is recommended for controlling a maximum of 64K devices. Some mainframe sites may require several SMC instances to provide management coverage for the entire storage pool. Each SMC instance, however, shares a server with other applications, and each instance remains quick, light, and independent.

SMC is intended to make array management faster and easier. Using dialog boxes structured into task wizards, SMC accelerates setup, configuration, and routine tasks. By providing simplified replication management and monitoring, SMC delivers ease of use that translates into efficient operation. Finally, managing for the future,
SMC will make new functionality available in the same simple intuitive manner, greatly lessening the learning curve necessary to implement any new technology and functionality.

With SMC, the mainframe user community now has an additional choice in Symmetrix array management. This choice is a tool that is easy to deploy, simplifies complex tasks through structured templates and wizards, delivers responsive user interaction, and readily integrates the tasks and changes of tomorrow.

Note: In May 2012, SMC merged into a new GUI product named Unisphere.

Symmetrix Performance Analyzer

EMC Symmetrix Performance Analyzer (SPA) is an intuitive, browser-based tool used to perform historical trending and analysis of Symmetrix array performance data. SPA was developed to work with the Symmetrix Management Console (SMC). The SPA interface can open in its own web window from the SMC menu or on its own. SPA adds an optional layer of data collection, analysis, and presentation tools to the SMC implementation. You can use SPA to:

◆ Set performance thresholds and alerts
◆ View high-frequency metrics as they become available
◆ Perform root cause analysis
◆ View graphs detailing system performance
◆ Drill down through data to investigate issues
◆ Monitor performance and capacity over time

SPA also provides a fast lane to display possible performance road blocks with one click, and includes export and print capability for all data graphs.

Note: In May 2012, SMC merged into a new GUI product named Unisphere.
Unisphere for VMAX

Available since May 2012, EMC Unisphere for VMAX replaces Symmetrix Management Console (SMC) and Symmetrix Performance Analyzer (SPA). With Unisphere for VMAX, customers can provision, manage, monitor, and analyze VMAX arrays from one console, significantly reducing storage administration time.

Unisphere for VMAX offers big-button navigation and streamlined operations to simplify and reduce the time required to manage data center storage. Unisphere for VMAX simplifies storage management under a common framework. You can use Unisphere to:

◆ Perform configuration operations
◆ Manage volumes
◆ Perform and monitor local and remote replication functions
◆ Monitor VMAX alerts
◆ Manage Fully Automated Storage Tiering (FAST and FAST/ VP)
◆ Manage user accounts and their roles

Unisphere for VMAX provides a single GUI interface for centralized management of your entire VMAX storage environment. This includes:

Configuration—Volume creation, set VMAX and volume attributes, set port flags, and create SAVE volume pools. Change volume configuration, set volume status, and create/dissolve metavolumes.

Performance—Performance data previously available through Symmetrix Performance Analyzer (SPA) is now included in Unisphere for VMAX. Monitor, analyze, and manage performance settings, such as threshold, alerts, metrics, and reports.

Replication monitoring—View and manage TimeFinder sessions. Includes session controls, details and modes. View and manage SRDF groups and pools.

Usability—Provides a simple, intuitive graphical user interface for array discovery, monitoring, configuration, and control of Symmetrix subsystems. A “single pane of glass” provides a view of Symmetrix arrays. From a dashboard view (that provides an overall view of the VMAX environment), you can drill down into the array dashboard. The array dashboard provides a summary of the physical and virtual
Virtual Provisioning

The Enginuity 5876 release for Symmetrix VMAX delivers Virtual Provisioning™ for CKD devices. After several years of successful deployment in open systems (FBA) environments, mainframe VMAX users now have the opportunity to deploy thin devices for DB2 and other mainframe applications. Virtual Provisioning for DB2 subsystems is described in detail in Chapter 3.

Fully Automated Storage Tiering

The Enginuity 5874 service release for Symmetrix VMAX systems provides the mainframe user Fully Automatic Storage Tiering (FAST) capability. Using FAST, whole DB2 devices can be moved from one storage type to another storage type based on user-defined policies. Device movement plans are automatically recommended (and optionally executed) by the FAST controller in the storage array. FAST operates at a granularity of a Symmetrix logical device, so it is always a full device (or metadevice) that is migrated between storage types (drive types and RAID protections).

Enginuity 5876 provides FAST functionality at the sub-volume and sub-dataset level. When a DB2 subsystem has been built using thin devices using Virtual Provisioning, it can take advantage of this feature. The 5876 Enginuity feature is called Fully Automated Storage Tiering Virtual Pools (FAST VP). This feature is described in more detail in Chapter 3.

capacity for that array. Additionally, the alerts are summarized here, and a user can further drill down into the alerts to get more information.

Security—Unisphere for VMAX supports the following types of authentication: Windows, LDAP, and local Unisphere users. User authentication and authorization using defined user roles, such as StorageAdmin, Auditor, SecurityAdmin, and Monitor.

Note: The first version of Unisphere was released in May 2012 and now contains the functionality of both SMC and SPA.
The Unisphere graphical user interface (GUI) is used to manage FAST and FAST VP operations. At this time, there is no specific batch job control to automate the functions of data movement.

### Data at Rest Encryption

Enginuity 5875 provides support for Symmetrix Data at Rest Encryption (Data Encryption) on all drive types supported by the Symmetrix VMAX system. The Data Encryption feature is available for new VMAX systems shipping with Enginuity 5875. It is not available as an upgrade option for existing VMAX systems (installed prior to the availability of Enginuity 5875), nor can it be made available through the RPQ process. This feature makes use of some special new hardware at the device adapter (DA) level. Further, this capability at the DA level to provide encryption for all drives controlled by that director represents a competitive advantage since other enterprise-class subsystems require that special drives be used to achieve encryption.

There are no user controls to enable or disable Data Encryption in a VMAX subsystem.

DARE adds Data Encryption capability to the back end of the VMAX array, and adds encryption key management services to the service processor. Key management is provided by the RSA® Key Manager (RKM) “lite” version that can be installed on the VMAX service processor by the Enginuity installer program. This new functionality for the service processor consists of the RKM client and embedded server software from RSA.
This chapter describes the advanced methods of provisioning VMAX storage for DB2 for z/OS subsystems.

- Virtual Provisioning................................................................. 68
- Fully Automated Storage Tiering ................................................ 82
Virtual Provisioning

Enginuity 5876 includes significant enhancements for mainframe users of the Symmetrix VMAX array that rival in importance to the original introduction of the first Symmetrix Integrated Cached Disk Array in the early 1990s. After several years of successful deployment in open systems (FBA) environments, mainframe VMAX users now have the opportunity to deploy Virtual Provisioning and Fully Automated Storage Tiering for Virtual Pools (FAST VP) for count key data (CKD) volumes.

This chapter describes the considerations for deploying a DB2 for z/OS database using Virtual Provisioning. An understanding of the principles that are described here will allow the reader to deploy DB2 for z/OS databases on thin devices in the most effective manner.

Terminology

The Virtual Provisioning for mainframe feature brings with it some new terms that may be unfamiliar to mainframe practitioners. Table 1 describes these new terms that are used extensively throughout this chapter.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>A logical unit of storage defined within a Symmetrix array.</td>
</tr>
<tr>
<td>Device capacity</td>
<td>The actual storage capacity of a device.</td>
</tr>
<tr>
<td>Device extent</td>
<td>The size of the smallest contiguous region of a device for which an extent mapping can occur.</td>
</tr>
<tr>
<td>Host-accessible device</td>
<td>A device that is presented on a FICON channel for host use.</td>
</tr>
<tr>
<td>Internal device</td>
<td>A device used for internal function of the array.</td>
</tr>
<tr>
<td>Storage pool</td>
<td>A collection of internal devices for some specific purpose.</td>
</tr>
<tr>
<td>Thin device</td>
<td>A host-accessible device that has no storage directly associated with it.</td>
</tr>
<tr>
<td>Data device</td>
<td>An internal device that provides storage capacity to be used by a thin device.</td>
</tr>
</tbody>
</table>
**Table 1** Virtual Provisioning terms (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extent mapping</td>
<td>Specifies the relationship between the thin device and data device extents. The extent sizes between a thin device and a data device do not need to be the same.</td>
</tr>
<tr>
<td>Thin pool</td>
<td>A collection of data devices that provides storage capacity for thin devices.</td>
</tr>
<tr>
<td>Thin pool capacity</td>
<td>The sum of the capacities of the member data devices.</td>
</tr>
<tr>
<td>Bind</td>
<td>The process by which one or more thin devices are associated to a thin pool.</td>
</tr>
<tr>
<td>Unbind</td>
<td>The process by which a thin device is disassociated from a given thin pool. When unbound, all previous extent allocations from the data devices are erased and returned for reuse.</td>
</tr>
<tr>
<td>Enabled data device</td>
<td>A data device belonging to a thin pool on which extents can be allocated for thin devices bound to that thin pool.</td>
</tr>
<tr>
<td>Disabled data device</td>
<td>A data device belonging to a thin pool from which capacity cannot be allocated for thin devices. This state is under user control. If a data device has existing extent allocations when a disable operation is executed against it, the extents are relocated to other enabled data devices with available free space within the thin pool.</td>
</tr>
<tr>
<td>Thin pool enabled capacity</td>
<td>The sum of the capacities of enabled data devices belonging to a thin pool.</td>
</tr>
<tr>
<td>Thin pool allocated capacity</td>
<td>A subset of thin-pool-enabled capacity that has been allocated for the exclusive use of all thin devices bound to that thin pool.</td>
</tr>
<tr>
<td>Thin pool pre-allocated capacity</td>
<td>The initial amount of capacity that is allocated when a thin device is bound to a thin pool. This property is under user control. For CKD thin volumes, the amount of pre-allocation is either 0% or 100%.</td>
</tr>
<tr>
<td>Thin device minimum pre-allocated capacity</td>
<td>The minimum amount of capacity that is pre-allocated to a thin device when it is bound to a thin pool. This property is not under user control.</td>
</tr>
</tbody>
</table>
Virtual Provisioning brings a new type of device into the mainframe environment called a thin device. Symmetrix thin devices are logical devices that can be used in many of the same ways that standard Symmetrix devices have traditionally been used. Unlike traditional Symmetrix devices, thin devices do not need to have physical storage completely allocated at the time the device is created and presented to a host. The physical storage that is used to supply disk space to thin devices comes from a shared storage pool called a thin pool. The thin pool is comprised of devices called data devices that provide the actual physical storage to support the thin device allocations.

Virtual Provisioning brings many benefits:

- Balanced performance
- Better capacity utilization
- Ease of provisioning
- Simplified storage layouts
- Simplified table space allocation
- Foundation for storage tiering

It is possible to over-provision thin storage volumes using Virtual Provisioning. Over-provisioning means that more space is presented to the mainframe host than is actually available in the underlying storage pools. Customers may decide to over-provision storage because it simplifies the provisioning process, or because they want to improve capacity utilization, or both. In either case, when an over-provisioning configuration is being used, careful monitoring of

---

### Table 1
Virtual Provisioning terms (continued)

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thin device written capacity</td>
<td>The capacity on a thin device that was written to by a host. In most implementations, this is a subset of the thin device allocated capacity.</td>
</tr>
<tr>
<td>Thin device subscribed capacity</td>
<td>The total capacity that a thin device is entitled to withdraw from a thin pool, which may be equal to or less than the thin device capacity.</td>
</tr>
<tr>
<td>Thin device allocation limit</td>
<td>The capacity limit that a thin device is entitled to withdraw from a thin pool, which may be equal to or less than the thin device subscribed capacity.</td>
</tr>
</tbody>
</table>
the storage pools is required to ensure that they do not reach 100 percent utilization. In addition, processes to reclaim storage that has been used and then deleted need to be executed to reclaim capacity and return it to the pool.

If over-provisioning is not needed, customers simply provide thin device capacity that matches the pool capacity. With this kind of implementation, neither pool monitoring nor space reclamation are necessary.

**Virtual Provisioning operations**

When a write is performed to a part of the thin device for which physical storage has not yet been allocated, the Symmetrix subsystem allocates physical storage from the thin pool only for that portion of the thin device. The Symmetrix operating environment, Enginuity, satisfies the requirement by providing a block of storage from the thin pool, called a thin device extent. This is the minimum amount of physical storage that can be reserved at a time for the dedicated use of a thin device. This approach reduces the amount of storage that is actually consumed.

The entire thin device extent is physically allocated to the thin device at the time the thin storage allocation is made. The thin device extent is allocated from any one of the data devices in the associated thin pool.

When a read is performed on a thin device, the data being read is retrieved from the appropriate data device in the thin pool to which the thin device is associated. If, for some reason, a read is performed against an unallocated portion of the thin device, standard record 0 is returned.

When more physical data storage is required to service existing or future thin devices (for example, when a thin pool is approaching maximum usage), data devices can be added dynamically to existing thin pools without the need for a system outage. A rebalance process can be executed to ensure that the pool capacities are balanced to provide an even distribution across all the devices in the pool, both new and old. New thin devices can also be created and associated with existing thin pools.

When data devices are added to a thin pool, they can be in an enabled or disabled state. In order for the data device to be used for thin extent allocation, it needs to be in the enabled state. For it to be
removed from the thin pool, it needs to be in the disabled state. Disabling a data device with active thin device allocations is the first step to removing the device from the pool. The second step is to drain it, which causes all the active extents on the device to be transferred to other enabled devices in the pool. When fully drained, the data device can be removed from the pool and utilized for other purposes.

Figure 15 depicts the relationships between thin devices and their associated thin pools. The host-visible thin devices are presented on a channel and have UCB addresses. The storage for the thin devices is provisioned from the thin device pool. The data devices in the thin device pool are created from RAID ranks in Symmetrix VMAX arrays. The data devices do not have an address on the channel.

The way thin extents are allocated across the data devices results in a form of striping in the thin pool. The more data devices that are in the thin pool, the wider the striping, and the greater the number of devices that can participate in application input/output (I/O). The thin extent size for CKD devices is 12 3390 tracks in size.

Requirements

Virtual Provisioning for CKD devices requires Enginuity code level of 5876 or later. In order to create thin pools and manage them, Mainframe Enablers V7.4 or later is required. Unisphere can also be used to manage Virtual Provisioning components. Thin device pools can only be created by the user, and they cannot be created during the
bin file (VMAX configuration) creation process. Thin pools cannot use RAID 10 protection when using CKD format. RAID 1, RAID 5, or RAID 6 are available for devices in the thin pool.

Host allocation considerations

Host dataset and volume management activities behave in subtly different ways when using Virtual Provisioning. These behaviors are usually invisible to end users. However, the introduction of thin devices can put these activities in a new perspective. For example, the amount of consumed space on a volume, as seen through ISPF 3.4 may be different from the actual consumed space from the thin pool. From a Virtual Provisioning point of view, a thin device could be using a substantial number of thin extents, even though the space is showing as available to the operating system. Alternatively, the volume may look completely full from a VTOC point of view and yet may be consuming minimal space from the thin pool.

These situations and perspectives are new with Virtual Provisioning. A clear understanding of them is necessary for a successful deployment of a thinly provisioned DB2 subsystem. An awareness of how DB2 and z/OS behave with this technology can lead to an educated choice of host deployment options and can yield maximum value from the Virtual Provisioning infrastructure.

Balanced configurations

With Virtual Provisioning on CKD volumes, it is now possible to have the nirvana of storage layouts: Balanced capacity and balanced performance. To achieve this goal prior to this product was a near impossibility. SMS does its level best to balance capacity, but without regard for performance. This manifests in the terribly skewed configurations that EMC encounters on a daily basis, where 20 percent of the DB2 volumes are performing 80 percent of the workload.

Deploying a skewed DB2 subsystem on thin devices still leaves the access to the thin devices skewed, but the storage layer is completely balanced due to the nature of the wide striping of the thin volumes across the pool. So now it is possible to have the best of both worlds: Better utilization and better performance, since bottlenecks at the disk layer have been removed.
Pool management

Fully provisioning volumes

It is possible to fully provision a z/OS volume when binding it to a thin pool. This means that all the tracks of the thin volume are allocated immediately from the thin pool. When this approach is used for all thin volumes assigned to a thin pool, and when there is more space in the pool than the aggregate total of thin-device capacity, it is a strategy to avoid oversubscription. This method of provisioning requires that the capacity of the thin pool is at least as large as the sum of the capacities of all the thin devices bound to the pool.

Fully provisioning thin volumes has the significant advantage of wide striping the thin volumes across all the devices in the thin pool (a benefit that should be utilized with large volumes, such as MOD27s, MOD54s, and EAVs).

To prevent space reclamation routines from returning free space that results from deleting files, the thin devices must be bound to the pool using the persistent option.

Oversubscription

Oversubscription is a key value of EMC Virtual Provisioning. Oversubscription allows storage administrators to provision more storage to the end user than is actually present. It is a common practice to request more storage than is actually needed simply to avoid the administrative overhead of the requisition/provisioning process. In many cases, this additional storage is never used. When this is the case, oversubscription can reduce your overall Total Cost of Ownership (TCO).

When implementing DB2 systems with oversubscription, it is very important to manage the capacity of a thin pool. As the thin pool starts to fill up, more storage has to be added to the pool to alleviate the situation. If a pool fills up and there is no more space to allocate to a write, DB2 receives I/O errors on the table space. This is not a desirable situation.

Messages are written to SYSLOG indicating that the thin pool utilization level after certain pre-defined levels have been exceeded. Automation can be used to track these messages and issue the appropriate alerts.
Adding storage to a pool and rebalancing

When the pool thresholds are exceeded (or at any other time), data devices may be added to the thin pool to increase capacity. When devices are added, an imbalance is created in the device capacity utilization. Some volumes may be nearly full, while the new volumes will be empty.

Figure 16 shows the four volumes that comprise a thin pool, each at 75 percent of capacity:

![Figure 16 Thin device pool with devices filling up](image)

Adding two additional volumes to the pool, and rebalancing the capacity, as shown in Figure 17, results in a capacity redistribution:

![Figure 17 Thin device pool rebalanced with new devices](image)

Note that the rebalancing activity is transparent to the host and executes without using any host I/O or CPU. For best practices, it is recommended that you do not add just two volumes to the pool. You should add enough devices to avoid frequent rebalancing efforts.

Space reclamation

Space reclamation is a process of returning disk space to the pool after it is no longer in use. For example, when a DB2 online REORG takes place, a duplicate copy of a table space is created. The original location of the dataset is eventually deleted. Even though it has been deleted from a z/OS perspective, and the space shows as available when viewing the VTOC, the space is actually still consumed from the thin pool.
The management of the space-reclaim function is not usually in the domain of the DB2 DBA, however, it is still instructive to understand how it works.

Space reclamation is performed by the Thin Reclaim Utility (TRU), which is a part of Mainframe Enablers V7.4. For thin devices that are not bound with the PERSIST and PREALLOCATE attributes, TRU enables the reclamation of the thin device track groups for reuse within the virtual pool by other thin devices. It does this by first identifying the free space in VTOC, initially by way of a scan function, then on an ongoing basis by way of the z/OS scratch exit. It then periodically performs a reclaim operation, which marks tracks as empty in the array (no user records, only standard R0). The Symmetrix space reclaim background task then returns these empty track groups to the free list in the virtual pool.

Note that this function only applies to volumes that have been thin provisioned in the Symmetrix subsystem. Volumes that are fully pre-allocated and marked persistent are not eligible for reclamation.

### Thin device monitoring

In a situation where over-provisioning has been used, that is to say the aggregate size of the thin devices exceeds the size of the thin pool, it is mandatory that some kind of pool monitor is active to alert on pool threshold conditions. The Symmetrix Control Facility actively monitors thin pools and sends alerts to SYSLOG based on certain percentage-full conditions of the thin pool. Based on these messages the storage administrator can add more devices to the thin pool as needed.

### Considerations for DB2 components on thin devices

There are certain aspects of deploying a DB2 subsystem using Virtual Provisioning that the DB2 DBA should be aware of. How the various components work in this kind of configuration depends on the component itself, its usage, and, sometimes, the underlying host data structure supporting the component. This section describes how the various DB2 components interact with Virtual Provisioning devices.
Active log files

Active log files are formatted by the DBA as a part of the subsystem creation process. Every single page of the log files is written to at this time, meaning that the log files become fully provisioned when they are initialized and will not cause any thin extent allocations after this. One thing to remember is that the active log files become striped across all the devices in the thin pool. Therefore, no single physical disk incurs the overhead of all the writes that come to the logs. If the standards for a particular site require that logs need to be separated from the data, then this can be achieved by creating a second thin pool that is dedicated for the active log files.

For better DB2 performance, it is always recommended to VSAM stripe the DB2 active log files. This recommendation holds true even if the DB2 active logs are deployed on thin devices.

Table space allocation

When volumes are not fully provisioned, that is they have no space pre-allocated to them, space is provided on demand from the thin pool. In this situation, a track written to for the first time causes space to be assigned from the thin pool. It is important to understand how DB2 writes to space from the table space if volumes are not pre-allocated.

DB2 V8 table space creation

When DB2 V8 allocates a table space with a PRIQTY value of greater than two cylinders, exactly two cylinders of the table space are formatted and written to by DB2. Among other things written to the table space are the page set headers and the table space space map.

DB2 V9 and V10 table space creation

When DB2 V9 or V10 creates a table space (not including hash), and if the PRIQTY is greater than 16 cylinders, 16 cylinders are written to with metadata and low-values. If the table space is less than 16 cylinders, all of it is written.

Hash table spaces

DB2 Version 10 introduces the hash table space capability. Hash table spaces avoid the overhead of having to traverse an index B-tree to locate the required data in a table when single row access is needed. Hash table spaces have yet to be employed widely by DB2 installations.
When tables are created in a hash table space, every single DB2 page is written to. This means that they are not very thin-friendly. If you overestimate how much space the hash table space will consume, the whole amount will be assigned from the thin pool, regardless of how much is actually being used.

**Temporary table spaces**

The DSNDB07 database contains all the temporary table spaces for all page sizes. By their nature, the table spaces are created and deleted. Over time, thin volumes that contain temporary data will get fully assigned storage from the pool, even if it seems (from the VTOC) that the volume is largely empty.

Of course, space reclamation processes can be run on these volumes to free up the space that is consumed in thin pool but not seen as allocated by the operating system. Ultimately, the reclamation process has to be run regularly because the condition will reoccur.

In the case of DSNDB07, it is recommended to either not run space reclamation against the volumes holding temp data, or to create the volumes that contain temp data on pre-allocated, persistent thin devices.

**PRIQTY and SECQTY**

When creating a table space, it is common to estimate the expected size of a primary allocation, as well as for the secondary allocation. Usually, the DBA tries to approximate the application requirements, though sometimes that is difficult to predict, particularly for green-field applications.

When deploying DB2 subsystems on thin devices, the DBA no longer needs to be that specific because over-allocation of space is not an issue. In fact, EMC recommends using PRIQTY -1 on the CREATE TABLESPACE statement, which allows DB2 to compute the correct space requirements using a sliding scale as the table space grows. Alternatively, the DBA can set the TSQTY DSNZPARM to the average size of a table space in the system, and not set anything at on the CREATE TABLESPACE statement.

**Replication considerations**

Thin devices behave in exactly the same way as normal, standard Symmetrix devices in regards to both local and remote replication. Thus, a thin device can be either a source or a target volume for a
TimeFinder replication process, and it can be either an R1 or an R2 in an SRDF configuration. There are no specific considerations related to thin devices using either of these two replication functions.

Migration from thick to thin

When a customer decides to use Virtual Provisioning for the first time in a mainframe environment, and needs to migrate existing DB2 subsystems to thin devices, a question arises: What is the best way to migrate from thick to thin?

Table 2 and Table 3 show some different utilities that can be used and the particular considerations with those method.

Table 2  Thick-to-thin host migrations

<table>
<thead>
<tr>
<th>Host-based copy</th>
<th>Disruptive to application (access must be interrupted)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFDSS, FDRDSF, other utilities</td>
<td>YES</td>
<td>Offers dataset extent consolidation</td>
</tr>
<tr>
<td>DFSMS reallocation</td>
<td>YES</td>
<td>Redefinition of batch datasets results in migration to a new SMS group of thin devices with volume selection changes in ACS routines</td>
</tr>
<tr>
<td>EMC z/OS Migrator</td>
<td>NO</td>
<td>Volume and dataset level in single product; smaller to larger volume REFVTTOC performed</td>
</tr>
<tr>
<td>TDMF (ZDMF)</td>
<td>NO</td>
<td>TDMF=Volume level product; ZDMF=dataset level product; smaller-to-larger volume REFVTTOC performed</td>
</tr>
<tr>
<td>FDRPAS (FDRMOVE)</td>
<td>NO</td>
<td>FDRPAS=Volume level product; FDRMOVE=dataset level product; smaller to larger volume REFVTTOC performed</td>
</tr>
</tbody>
</table>
All of these methods copy DB2 datasets in their entirety, including the embedded empty space due to over-allocation of the dataset. At a time when an outage is allowed, DB2 should be brought down to free up the unused tracks of the linear table datasets that are known to be over allocated. This can be done by using the ADRDSSU program as in the following JCL example:

```
//STEP01 EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
  RELEASE INCLUDE (DBR1.DSNDBC.dbname.**)
/*
This example releases all the allocated but unused space in the linear datasets for the database dbname, referenced by the INCLUDE statement. The RELEASE command does not release space from guaranteed-space VSAM extended-format datasets.

Note that this process does not return the unused space to the thin pool, but merely gets the VTOC to accurately reflect the amount of used space in the table space. To return the space back to the pool to make it available for other devices, reclamation must be performed on the volume as described earlier.

---

**Performance considerations**

Thin provisioning provides balanced performance due to the wide-striping and balanced capacity in the thin pool. With the small chunk size of 12 tracks, it is highly unlikely that any workload skew will be visible on a particular disk. This drives the performance capability of the system up to much higher levels than can be achieved on normally provisioned systems.
There are some other considerations that need to be understood when deploying DB2 using Virtual Provisioning.

**Chunk size and sequential processing**

The Virtual Provisioning chunk size of 12 tracks is the effective stripe size across the thin pool. This allows for more balanced performance as mentioned above. One effect of this striping is to make DB2 pages that are adjacent on the thin device to be non-adjacent on the actual physical disks. This has the potential to de-optimize DB2 sequential table space scans on the actual array because the disk is unable to go into streaming mode. A single user performing a sequential scan on a thick device will complete the scan faster than the equivalent scan on a thin device.

However, it is very important to understand, that it is extremely rare to have a single user/process owning a disk. On busy systems, the disk is going to be shared among many users and processes, and the disk/head mechanism will be servicing many different access extents on these very large physical disks. This disables the ability of the disk to stream reads to the host. Thus, the random layout of the chunks on the DASD is not such a big problem in busy environments.

**Virtual provisioning overhead**

When a block is read from or written to a thin device, Enginuity examines the metadata (pointers) for the thin device to determine the location of the appropriate block in the thin pool. There is a slight cost in I/O response time for this activity. In addition, when a new chunk is allocated from the thin pool, there is also additional code that needs to be executed. Both of these overheads are extremely small when compared to the overall performance of a widely striped DB2 subsystem that runs with no disk-storage bottlenecks.
Fully Automated Storage Tiering

This section provides information for DB2 for z/OS and FAST VP deployments and also includes some best practices regarding implementation of DB2 for z/OS with FAST VP configurations.

Introduction

FAST VP is a dynamic storage tiering solution for the VMAX Family of storage controllers that manages the movement of data between tiers of storage to maximize performance and reduce cost. Volumes that are managed by FAST VP must be thin devices.

Delivered with Enginuity 5876, FAST VP is a VMAX array feature that dynamically moves data between tiers to maximize performance and reduce cost. It non-disruptively moves sets of 10 track groups (6.8 MB) between storage tiers automatically at the sub-volume level in response to changing workloads. It is based on, and requires, virtually provisioned volumes in the VMAX array.

EMC determined the ideal chunk size (6.8 MB) from analysis of 50 billion I/Os provided to EMC by customers. A smaller size increases the management overhead to an unacceptable level. A larger size increases the waste of valuable and expensive Enterprise Flash drive (EFD) space by moving data to EFD that is not active. Tiering solutions using larger chunk sizes require a larger capacity of solid-state drives, which increases the overall cost.

FAST VP fills a long-standing need in z/OS storage management: Active performance management of data at the array level. It does this very effectively by moving data in small units, making it both responsive to the workload and efficient in its use of control-unit resources.

Such sub-volume, and more importantly, sub-dataset, performance management has never been available before and represents a revolutionary step forward by providing truly autonomic storage management.

As a result of this innovative approach, compared to an all-Fibre Channel (FC) disk drive configuration, FAST VP can offer better performance at the same cost, or the same performance at a lower cost.
FAST VP also helps users reduce DASD costs by enabling exploitation of very high capacity SATA technology for low-access data, without requiring intensive performance management by storage administrators.

Most impressively, FAST VP delivers all these benefits without using any host resources whatsoever.

FAST VP uses three constructs to achieve this:

**FAST storage group**
A collection of thin volumes that represent an application or workload. These can be based on SMS storage group definitions in a z/OS environment.

**FAST policy**
The FAST VP policy contains rules that govern how much capacity of a storage group (in percentage terms) is allowed to be moved into each tier. The percentages in a policy must total at least 100 percent, but may exceed 100 percent. This may seem counter-intuitive but is easily explained. Suppose you have an application that you want FAST VP to determine exactly where the data needs to be without constraints, you would create a policy that permits 100 percent of the storage group to be on EFD, 100 percent on FC, and 100 percent on SATA. This policy totals 300 percent. This kind of policy is the least restrictive that you can make. Mostly likely you will constrain how much EFD and FC a particular application is able to use but leave SATA at 100 percent for inactive data.

Each FAST storage group is associated with a single FAST policy definition.

**FAST tier**
A collection of up to four virtual pools with common drive technology and RAID protection. At the time of writing, the VMAX array supports four FAST tiers.

Figure 18 depicts the relationship between VP and FAST VP in the VMAX arrays. Thin devices are grouped together into storage groups. Each storage group is usually mapped to one or more applications or DB2 subsystems that have common performance characteristics.
A policy is assigned to the storage group that denotes how much of each storage tier the application is permitted to use. The figure shows two DB2 subsystems, DB2A and DB2B, with a different policy.

**DB2A**
This has a policy labeled Optimization, which allows DB2A to have its storage occupy up to 100 percent of the three assigned tiers. In other words, there is no restriction on where the storage for DB2A can reside.

**DB2B**
This has a policy labeled Custom, which forces an exact amount of storage for each tier. This is the most restrictive kind of policy that can be used and is effected by making the total of the allocations equal one hundred percent.

More details on FAST VP can be found in the white paper *Implementing Fully Automated Storage Tiering for Virtual Pools (FAST VP) for EMC Symmetrix VMAX Series Arrays*.

---

**Best practices for DB2 and FAST VP**

In this section, some best practices are presented for DB2 for z/OS in a FAST VP context. DB2 can automatically take advantage of the advanced dynamic and automatic tiering provided by FAST VP without any changes. However, there are some decisions that need to be made at setup time with respect to the performance and capacity requirements on each tier. There is also the setup of the storage
group, as well as the time windows, and some other additional parameters. All of the settings can be performed using Unisphere for VMAX.

**Unisphere for VMAX**

Unisphere for VMAX is used to manage all the necessary components to enable FAST VP for DB2 subsystems. While details on the use of Unisphere are beyond the scope of this TechBook, the following parameters need to be understood to make an informed decision about the FAST VP setup.

**Storage groups**

When creating a FAST VP storage group (not to be confused with an SMS storage group), you should select thin volumes that are going to be treated in the same way, with the same performance and capacity characteristics. A single DB2 subsystem and all of its volumes might be an appropriate grouping. It might also be convenient to map a FAST VP storage group to a single SMS storage group, or you could place multiple SMS storage groups into one FAST VP storage group. Whatever is the choice, remember that a FAST VP storage group can only have thin devices in it.

If you have implemented Virtual Provisioning and are later adding FAST VP, when creating the FAST VP storage group with Unisphere, you must use the option Manual Selection and select the thin volumes that are to be in the FAST VP storage group.

**FAST VP policies**

For each storage group that you define for DB2, you need to assign a policy for the tiers that the storage is permitted to reside on. If your tiers are EFD, FC, and SATA, as an example, you can have a policy that permits up to 5 percent of the storage group to reside on EFD, up to 60 percent to reside on FC, and up to 100 percent to reside on SATA. If you don’t know what proportions are appropriate, you can use an empirical approach and start incrementally. The initial settings for this would be 100 percent on FC and nothing on the other two tiers. With these settings, all the data remains on FC (presuming it lives on there already). At a later time, you can dynamically change the policy to add the other tiers and gradually increase the amount of capacity allowed on EFD and SATA. This can be performed using the
Unisphere GUI. Evaluation of performance lets you know how successful the adjustments were, and the percentage thresholds can be modified accordingly.

A policy totaling exactly 100 percent for all tiers is the most restrictive policy and determines what exact capacity is allowed on each tier. The least restrictive policy allows up to 100 percent of the storage group to be allocated on each tier.

DB2 test systems would be good targets for placing large quantities on SATA. This is because the data can remain for long times between development cycles, and the performance requirements can be somewhat looser. In addition, test systems do not normally have a high performance requirement and most likely will not need to reside on the EFD tier. An example of this kind of policy would be 50 percent on FC and 100 percent on SATA.

Even with high I/O rate DB2 subsystems, there is always data that is rarely accessed that could reside on SATA drives without incurring a performance penalty. For this reason, you should consider putting SATA drives in your production policy. FAST VP will not demote any data to SATA that is accessed frequently. An example of a policy for this kind of subsystem would be 5 percent on EFD, 100 percent on FC, and 100 percent on SATA.

**Time windows for data collection**

Make sure that you collect data only during the times that are critical for the DB2 applications. For instance, if you REORG table spaces on a Sunday afternoon, you may want to exclude that time from the FAST VP statistics collection. Note that the performance time windows apply to the entire VMAX controller, so you need to coordinate the collection time windows with your storage administrator.

**Time windows for data movement**

Make sure you create the time windows that define when data can be moved from tier to tier. Data movements can be performance-based or policy-based. In either case, it places additional load on the VMAX array and should be performed at times when the application is less demanding. Note that the movement time windows apply to the entire VMAX controller, so you need to coordinate them with other applications requirements that are under FAST VP control.
DB2 active logs

Active log files are formatted by the DBA as a part of the subsystem creation process. Every single page of the log files is written to at this time, meaning that the log files become fully provisioned when they are initialized and will not cause any thin extent allocations after this. The DB2 active logs are thus spread across the pool and incur the benefit of being widely striped.

FAST VP does not use cache hits as a part of the analysis algorithms to determine what data needs to be moved. Since all writes are cache hits, and the DB2 log activity is primarily writes, it is highly unlikely that FAST VP will move parts of the active log to another tier. Think of it this way: Response times are already at memory speed due to the DASD fast write response, so can you make it any faster?

For better DB2 performance, it is recommended to VSAM stripe the DB2 active log files, especially when SRDF is being used. This recommendation holds true even if the DB2 active logs are deployed on thin devices.

DB2 REORGs

Online REORGs for DB2 table spaces can undo a lot of the good work that FAST has accomplished. Consider a table space that has been optimized by FAST VP and has its hot pages on EFD, its warm pages on FC, and its cold pages on SATA. At some point, the DBA decides to do an online REORG. A complete copy of the table space is made in new unoccupied space and potentially unallocated part of the thin storage pool. If the table space can fit, it is completely allocated on the thin pool associated with the new thin device containing the table space. This new table space on a thin device is (most likely) all on Fibre Channel drives again. In other words, de-optimized. After some operational time, FAST VP begins to promote and demote the table space track groups when it has obtained enough information about the processing characteristics of these new chunks. So, it is a reality, that a DB2 REORG could actually reduce the performance of the tables space/partition.

There is no real good answer to this. But on the bright side, it is entirely possible that the performance gain through using FAST VP could reduce the frequency of REORGs, if the reason for doing the REORG is performance based. So when utilizing FAST VP, you should consider revisiting the REORG operational process for DB2.
z/OS utilities

Any utility that moves a dataset/volume (for instance ADRDSSU) changes the performance characteristics of that dataset/volume until FAST VP has gained enough performance statistics to determine which track groups of the new dataset should be moved back to the different tiers they used to reside upon. This could take some time, depending on the settings for the time windows and performance collection windows.

DB2 and SMS storage groups

There is a natural congruence between SMS and FAST VP where storage groups are concerned. Customers group applications and databases together into a single SMS storage group when they have similar operational characteristics. If this storage group were built on thin devices (a requirement for FAST VP), a FAST VP storage group could be created to match the devices in the SMS storage group. While this is not a requirement with FAST VP, it is a simple and logical way to approach the creation of FAST VP storage groups. Built in this fashion, FAST VP can manage the performance characteristics of the underlying applications in much the same way that SMS manages the other aspects of the storage management.

DB2 and HSM

It is unusual to have HSM archive processes apply to production DB2 datasets, but it is fairly common to have them apply to test, development, and QA environments. HMIGRATE operations are fairly frequent in those configurations, releasing valuable storage for other purposes. With FAST VP, you can have the primary volumes augmented with economic SATA capacity and use less aggressive HSM migration policies.

The disadvantages of HSM are:

- When a single row is accessed from a migrated table space/partition, the entire dataset needs to be HRECALLed.
- When HSM migrates and recalls datasets, it uses costly host CPU and I/O resources.

The advantages of using FAST VP to move data to primary volumes on SATA are:
◆ If the dataset resides on SATA, it can be accessed directly from there without recalling the entire dataset.

◆ FAST VP uses the VMAX storage controller to move data between tiers.

An example of a FAST VP policy to use with DB2 test subsystems is 0 percent on EFD, 50 percent on FC, and 100 percent on SATA. Over time, if the subsystems are not used, and there is demand for the FC tier, FAST VP will move the idle data to SATA.
This chapter describes the various local replication techniques for copying DB2 objects and systems.

- DB2 object cloning ................................................................. 92
- DB2 system cloning ............................................................... 110
DB2 object cloning

Many DB2 customers require DB2 objects to be cloned across or within DB2 systems. The reasons for cloning DB2 objects include:

- Building test environments
- Providing system integration and test environments
- Building read-only decision support environments
- Satisfying a number of other business-related requirements for DB2 data

The amount of data cloned and the frequency at which it is cloned tend to be driven by the business requirements of the organization and ultimately determine the technology used to replicate the information.

Historically, DB2 systems have been cloned using application programs to unload databases and then reload the data back into the target databases, or using the DB2 utility DSN1COPY with OBIDXLAT and RESET parameters. Other data cloning implementations use Hierarchical Storage Management (HSM) software and data mover software, such as FDR or DFDSS, to copy DB2 datasets to tape, and then restore the datasets on a target database’s z/OS and DB2 system. Both of these data-cloning implementations require significant mainframe I/O and CPU resources to copy the data from its source to some intermediate storage device, and then to the target system.

DB2 systems are sometimes used for integration testing. These types of test environments tend to be created, manipulated by test programs, and then are reset to a base copy in an iterative manner allowing test programs to process on baseline versions of data.

Cloning data across and within DB2 instances using EMC solutions involves these separate steps:

1. Creating target metadata by creating objects on the target DB2.
2. Copying the data.
3. Resetting the level ID on the target.
4. Changing the DBID, PSID to match the target DB2 catalog.
The above steps can be divided into two categories: (1) metadata management (steps 1, 3, and 4), which is performed on the host, and (2) replicating the data (step 2), which is executed within the Symmetrix controller. A high-level view of cloning data is shown in Figure 19.

Figure 19  DB2 object cloning overview

Cloning DB2 data with EMC products minimizes mainframe resource consumption by moving data within the Symmetrix system, and not utilizing host I/O or CPU.

When objects are created in DB2, identifiers are assigned to them. The identifiers are kept in the DB2 directory and also in the DB2 catalog (and can be queried using a simple SQL statement). Some of them also reside in the dataset for the object.

The identifiers that are important for the object cloning process are:

- DBID of a database is kept in the header page of each page set.
- PSID of a table space is kept in the header page of each page set.
- ISOBID of an index (type 1 or type 2) is kept in the header page of each page set.
- OBID of a table is kept in each data page.
- Level ID is kept in the header page of each page set.

When a database is created in DB2, a DBD, which is the internal DB2 representation of the database, is created in the DB2 directory. An internal DB2 identifier (DBID) is then associated with this DBD. Each object that is created in that database is assigned a unique identifier within the DBD. All identifiers must be unique within a DBD, regardless of their type (OBID, PSID, ISOBID, and others).
Movement of data between two different DB2 databases requires that these identifiers be changed so that the target database can properly access and manipulate data for an object. If the object identifiers for the tables, databases, or page sets in the dataset do not match what is stored in the DB2 catalog and DB2 directory, DB2 detects the mismatch and full read/write access to the data is prevented.

DSN1COPY can modify and translate all these identifiers during the copy process, through the OBIDXLAT and RESET parameters. However, it requires that each data page be processed by the utility. Using DSN1COPY to change these values, negates the value of TimeFinder/Mirror or TimeFinder/Snap in that all the data needs to be processed by the utility, consuming significant CPU and I/O resources in the process.

EMC products can be used to copy the data and then make it available instantly by simply splitting BCVs or snapping associated datasets or volumes. TimeFinder/Mirror and TimeFinder/Snap cloning procedures require that the OBIDs and DBIDs of the source and targets be identical.

---

### Creating objects on the target DB2

Create the cloned tables in the target DB2 database specifying OBIDs that are the same as on the source. Refer to “REXX Example #1” on page 279 and “REXX Example #2” on page 286 for REXX examples provided to assist with this step. Use the following procedure to create the tables in the target DB2 database:

1. Acquire the OBIDs of the source tables using the following sample SQL:

   ```sql
   SELECT OBID FROM SYSIBM.SYSTABLES
   WHERE NAME = 'source_table_name' AND CREATOR = 'source_table_creator_name'
   ;
   ```

2. Specify the source OBIDs in the CREATE TABLE statements for the target tables, as shown in the following example:

   ```sql
   SET CURRENT SQLID='source table creator name'
   CREATE TABLE target_table
   (COL1 CHAR(10) NOT NULL,
    (COL2 CHAR(10) NOT NULL)
   IN target_db_name.target_ts_name
   OBID source_obid
   ;
   ```
The OBID is a pervasive identifier embedded in every page in a page set. Specifying the OBID on the target CREATE TABLE statement is easier, faster, and less resource consuming than attempting to change the OBID in every page to match the DB2 catalog definitions in the target DB2 instance. This procedure ensures the DB2 catalog table SYSIBM.SYSSTABLES and the DBD on the target system contain the same OBID as is stored in the actual pages. The only consideration in this step is that the OBID used in the CREATE TABLE statement cannot already be in use on the target database. If it is already used, there are two possible ways to get around this:

- Create the target table in a different database where the source table’s OBID is not yet being used.
- Drop and recreate the object on the source DB2 database using a different OBID that is not currently in use on either the source or target DB2 databases.

### Cloning the data

Copying the data can be done in three ways:

- TimeFinder/Clone—dataset snap
- TimeFinder/Mirror—Volume establish/split
- TimeFinder/Clone—Volume snap
- TimeFinder/Snap—Virtual device snap

The optimal technique to use depends on the density of source datasets on the volumes, the number of copies required, and the time the source tables can be unavailable for update. For more details on selecting the best method to use, refer to “Cloning method usage considerations” on page 108.

### Cloning DB2 objects using dataset snap

Cloning DB2 objects using dataset snap is best for DB2 objects that are sparsely populated across a number of volumes. The procedure for cloning objects using the SNAP DATASET command is slightly different from the volume-based replication in that the target volume metadata (VOL1 label, VTOC, VVDS) does not need to be reconditioned to rename and re-catalog the target datasets. The rename is done by the SNAP utility. Figure 20 on page 96 shows an example of cloning DB2 objects using dataset snap.
Cloning DB2 objects and systems

Figure 20 Cloning DB2 objects using TimeFinder dataset snap

To copy the data using dataset snap, use the following procedure, which refers to Figure 20:

1. Change source object to READ ONLY access to externalize source DB2 buffers to disk.

When the data movement is desired, the table space must be placed in READ ONLY mode by means of a START DATABASE command, followed by a DB2 QUIESCE utility with WRITE(YES) to ensure that all changed pages are externalized to disk by the buffer manager.

Use the following sample JCL to perform the quiesce:

```jcl
//DB2STA EXEC PGM=IKJEFT01
//SYSIN DD DUMMY
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD *
   DSN SYSTEM(source_ssid)
   -START DB(source_db_name) SPACE(source_ts_name)
      ACCESS(RO)
END
/*
/*
//** TABLE SPACE QUIESCE
//UTIL EXEC DSNUPROC,SYSTEM=source_ssid,
//UID='UTIL1',UTPROC='' 
//DSNUPROC.SYSTEM DD *
   QUIESCE TABLESPACE source_db_name.source_ts_name
      WRITE(YES)
/*
```

---

Host

ICO-IMG-000714

Source data

Symmetrix

TimeFinder/Snap

Data set

SNAP

Target data

Symmetrix
2. Stop objects on target DB2.

Use the DB2 STOP command to deallocate the cloned datasets from the target DB2 instance. Use the following sample JCL to perform the stop:

```
//DB2STO EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*   
//SYSIN DD DUMMY        
//SYSTSIN DD *          
   DSN SYSTEM(target_ssid)   
   -STOP DB(target_dbname) SPACE(target_tsnme)  
   END                      
/*
```

3. Snap the datasets associated with the objects to be cloned.

Use the following sample JCL to perform a dataset SNAP.

```
//RUNSNAP EXEC PGM=EMCSNAP
//SYSUDUMP DD SYSOUT=* 
//SYSSOUT DD SYSOUT=*  
//QCINPUT DD *          
SNAP DATASET (  
   SOURCE('srchlg.DSNDBC.dbname.tsnme.I0001.A001') -  
   TARGET('tgthlg.DSNDBC.dbname.tsnme.I0001.A001') -  
   VOLUME(EMCT01)-  
   REPLACE(Y)-  
   REUSE(Y)-  
   FORCE(N)-  
   HOSTCOPYMODE(SHR)-  
   DEBUG(OFF)-  
   TRACE(OFF))  
/*
```

4. Start source DB2 objects for normal processing. Use the following sample JCL to start the source DB2 table space.

```
//DB2STA EXEC PGM=IKJEFT01  
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*   
//SYSIN DD DUMMY        
//SYSTSIN DD *          
   DSN SYSTEM(source_ssid)   
   -START DB(source_db_name) SPACE(source_tsnme)  
   ACCESS(RW)                  
   END                        
/*
```

5. Start target DB2 objects for normal processing.
Cloning of DB2 Objects and Systems

Use the following sample JCL to start the target DB2 table space.

```bash
//DB2STA EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYIN DD DUMMY 
//SYSTSIN DD * 
  DSN SYSTEM(target_ssid) 
  -START DB(target_dbname) SPACE(target_tsnme) 
       ACCESS(RW) 
END
/*

6. Refer to “Resetting the level ID on the target” on page 104 for the procedure to reset the level ID on the target object.

7. Refer to “Matching the DBID/PSID to the target catalog” on page 105 to match the target DB2 catalog for the procedures to adjust the target object’s DBID and PSID.

Cloning DB2 objects using BCVs

TimeFinder/Mirror copies data from a full volume source device to a BCV. This is a space-equivalent copy. Objects to be cloned must already exist on the target database. The procedure for defining the target objects can be found in “Creating objects on the target DB2” on page 94. Figure 21 is an example of using TimeFinder/Mirror to copy DB2 objects present on standard volumes.

Figure 21  Object cloning using TimeFinder/Mirror

To clone DB2 objects using TimeFinder/Mirror, use the following procedure:
1. Stop objects on target DB2. Use the DB2 STOP command to deallocate the cloned datasets from the target DB2 instance.

Use the following sample JCL to perform the DB2 STOP command:

```
//DB2STO EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSSIN DD DUMMY  
//SYSIN DD *  
  DSN SYSTEM(target_ssid)  
  -STOP DB(target_dbname) SPACE(target_tsname)  
  END  
/*
```

2. Delete underlying VSAM datasets. The existing datasets must be deleted. These datasets were created either by DB2 when the CREATE TABLE was done or during a previous cloning operation.

3. Vary off the BCVs. This ensures there are no allocations on the volumes.

4. Perform a TimeFinder/Mirror ESTABLISH command to start the copying process from the source DB2 volumes that contain the table spaces and index spaces to be cloned to the BCVs. The ESTABLISH can be done while processing on the source DB2 continues and is transparent to applications running there. Use the following sample JCL to perform the ESTABLISH.

```
//STEP1 EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD * 
  GLOBAL WAIT,MAXRC=0,FASTEST(Y)  
  QUERY 1,5420  
  ESTABLISH 2,5420,5410  
  QUERY 3,5420  
/*
```

5. The table space must be placed in read-only mode by means of a START DATABASE command followed by a DB2 QUIESCE utility with WRITE(YES) to ensure that all changed pages are externalized to disk by the buffer manager.

Use the following sample JCL to perform the quiesce.

```
//DB2RO EXEC PGM=IKJEFT01  
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
```
Cloning of DB2 Objects and Systems

```sql
//SYSIN DD DUMMY
//SYSTSIN DD *
DSN SYSTEM(source_ssid)
-START DB(source_db_name) SPACE(source_ts_name)
   ACCESS(RO)
END
/*
/*/ 
/* TABLE SPACE QUIESCE
/UTIL EXEC DSNUPROC,
 // SYSTEM=source_ssid,
 // UID='UTIL1',UTPROC=''
//DSNUPROC.SYSIN DD *
   QUIESCE TABLESPACE source_db_name.source_ts_name
   WRITE(YES)
/*

6. Use the following sample JCL to perform the TimeFinder/Mirror SPLIT.

```sql
//STEP1 EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD *
GLOBAL NOWAIT,MAXRC=0
QUERY 1,5420
SPLIT 2,5420
QUERY 3,5420
/*

7. Start source DB2 objects for normal processing. Normal processing can now be resumed on the source objects.

Use the following JCL to perform the DB2 START command.

```sql
//DB2STA EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTIN DD *
//SYSTSIN DD *
DSN SYSTEM(source_ssid)
-START DB(source_db_name) SPACE(source_ts_name)
   ACCESS(RW)
END
/*

8. The BCV is now ready to be processed in order to make the table spaces and index spaces accessible by the target DB2 instance. The TimeFinder Utility is used to relabel the volumes, so that they can be varied online for dataset processing. The RELABEL statement specifies the new VOLSER. The RENAME statement changes the high-level qualifier of the datasets to match that of
the target DB2 instance. The `PROCESS` statement identifies the type of files to be processed for each BCV. The `CATALOG` statement identifies the ICF catalog that owns the renamed datasets.

Use the following sample JCL to condition the BCV:

```plaintext
//TFURNM EXEC PGM=EMCTFU
//SYSOUT DD SYSOUT=*  
//TFINPUT DD *  
   RELABEL CUU=5420,OLD-VOLSER=EMCS01,NEW-VOLSER=EMCT01  
   PROCESS CUU=5420,VOLSER=EMCT01,VSAM  
   CATALOG DB2TPRD1.USER.CATALOG,DEFAULT  
   RENAME source_hlq,target_hlq  
/*
```

9. Start target DB2 objects for normal processing.

Use the following sample JCL to perform the enable the target table spaces and index spaces for processing:

```plaintext
//DB2STA EXEC PGM=IKJEFT01  
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSTIN DD DUMMY  
//SYSTSIN DD *  
   DSN SYSTEM(target_ssid)  
   -START DB(target_dbname) SPACE(target_tsname) ACCESS(RW)  
   END  
/*
```

10. Refer to “Resetting the level ID on the target” on page 104 for the procedure to reset the level ID on the target object.

11. Refer to “Matching the DBID/PSID to the target catalog” on page 105 for the procedure to adjust the target object’s DBID and PSID.

**Cloning DB2 objects using full volume snap or virtual device snap**

The procedure of cloning using TimeFinder/Clone or TimeFinder/Snap is similar to that of TimeFinder/Mirror. Objects may be cloned using a full volume snap or a virtual device snap. Target objects should be created on the target DB2, and the source table spaces must be quiesced before doing the volume snap. Figure 22 on page 102 shows an example of cloning DB2 objects using TimeFinder volume snap.
Cloning of DB2 Objects and Systems

To clone DB2 objects using volume snap, use the following procedure:

1. Stop objects on target DB2. Use the DB2 STOP command to deallocate the cloned datasets from the target DB2 instance. Use the following sample JCL to perform the stop.

   ```
   //DB2STO EXEC PGM=IKJEFT01
   //SYSTSPRT DD SYSOUT=*  
   //SYSPRINT DD SYSOUT=*  
   //SYSIN DD DUMMY  
   //SYSTSIN DD *
   DSN SYSTEM(target_ssid)
   -STOP DB(target dbname) SPACE(target tsname)
   END
   /*
   ```

2. Delete underlying target VSAM datasets. Before renaming the copied datasets to the names that DB2 expects on the target, the existing datasets must be deleted. These datasets were created either by DB2 when the CREATE TABLESPACE was done or during a previous cloning operation.

3. The source table space must be placed in read-only mode using a START DATABASE command, followed by a DB2 QUIESCE utility with WRITE(YES) to ensure that all changed pages are externalized to disk by the buffer manager.

   Use the following sample JCL to perform the quiesce:

   ```
   //DB2STA EXEC PGM=IKJEFT01
   //SYSTSPRT DD SYSOUT=*  
   //SYSPRINT DD SYSOUT=*  
   ```
4. Snap the volumes containing the datasets that are associated with the objects being cloned. The following sample JCL can be used to perform a full-volume snap.

```jcl
//RUNSNAP EXEC PGM=EMCSNAP
//QCOUTPUT DD SYSOUT=*  //SYSOUT DD SYSOUT=*  //QCINPUT DD *
SNAP VOLUME (SOURCE(VOL(EMCS01)) TARGET(VOL(EMCT01)) REPLACE(YES))
/*
```

Alternately, the following sample JCL can be used to perform a volume SNAP to a virtual device.

```jcl
//RUNSNAP EXEC PGM=EMCSNAP
//SYSOUT DD SYSOUT=*  //QCOUTPUT DD SYSOUT=*  //QCINPUT DD *
SNAP VOLUME (SOURCE (UNIT(C400)) - NEWVOLID(EMCT01) - VDEV(FREE))
/*
```

Which volume copy method to use depends on several factors. Refer to “DB2 object cloning considerations” on page 106 for more details.

5. Resume normal processing on the source objects. Use the following sample JCL to perform the DB2 start.

```jcl
//DB2STA EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*  //SYSPRINT DD SYSOUT=*  //SYSIN DD DUMMY
//SYSTSIN DD *
DSN SYSTEM(src_ssid)
```
Cloning of DB2 Objects and Systems

```
-START DB(src_dbname) SPACE(src_tsname)
    ACCESS(RW)
END
/*

6. Rename copied datasets. Use the TimeFinder Utility to rename datasets. The following sample JCL renames the source HLQ to the target HLQ:

    //TFURNM EXEC PGM=EMCTFU
    //SYSOUT DD SYSOUT=*  
    //SYSIN DD *
    PROCESS CUU=5420,VOLSER=EMCT01,VSAM
    CATALOG DB2TPRD1.USER.CATALOG,DEFAULT
    RENAME src_hlq,tgt_hlq
    /*

7. Start target DB2 objects for normal processing.

Use the following sample JCL to perform the DB2 START:

    //DB2STA EXEC PGM=IKJEFT01
    //SYSTSPRT DD SYSOUT=*  
    //SYSPRINT DD SYSOUT=*  
    //SYSIN DD DUMMY
    //SYSTSIN DD *
    DSN SYSTEM(tgt_ssid)
    -START DB(tgt_dbname) SPACE(tgt_tsname) ACCESS(RW)
    END
    /*

8. Refer to “Resetting the level ID on the target” on page 104 for the procedure to reset the level ID on the target object.

9. Refer to “Matching the DBID/PSID to the target catalog” on page 105 for the procedure to adjust the target object's DBID and PSID.

Resetting the level ID on the target

Every page set in DB2 has a value in the header page called a level ID. This is also stored in the DB2 directory and is used to detect if a different version of a page set has been restored in error. The level ID is a log RBA value that is updated by open, close, and checkpoint processing for the table space. Because the cloned page set is from a source DB2 instance with different RBA values, the level ID will not match the target DB2 catalog. A reset level ID operation using the REPAIR utility must be performed to reset the level ID in the page set header page.
Use the following sample JCL to reset the level ID of the cloned objects on the target DB2.

```sql
//UTIL EXEC DSNUPROC,SYSTEM=tgt_ssid,
// UID='UTIL1',UPTPROC=''
//DSNUPROC.SYSIN DD *
  REPAIR LEVELID TABLESPACE tgt dbname.tgt tsname
/*
```

---

**Matching the DBID/PSID to the target catalog**

If target objects are to be updated, then the DBID and PSID must be repaired to match the values in the target DB2 catalog. DBID and PSID values in the target DB2 catalog do not match what is actually in the page sets since they are from the source DB2 instance. This mismatch can be remedied by running the DB2 REPAIR utility to alter these values in the page set headers to match the values in the target DB2 catalog.

---

**Determining DBID and PSID in the source and target DB2 systems**

Use the following sample SQL to determine DBID and PSID in the source and target DB2 instances:

For a table space:

```sql
SELECT HEX(DBID), HEX(PSID) FROM
SYSIBM.SYSTABLESPACE WHERE NAME = 'tsname'
AND DBNAME = 'dbname'
```

For an index space:

```sql
SELECT HEX(DBID), HEX(ISOBID) FROM SYSIBM.SYSINDEXES
WHERE NAME='indexname' AND DBNAME = 'dbname'
```

---

**Performing the repair**

Use the following sample JCL to perform the repair:

For a table space:

```sql
//REPAIR EXEC DSNUPROC,SYSTEM=tgt_ssid,
// UID='REPR',UPTPROC=''
//DSNUPROC.SYSIN DD *
REPAIR OBJECT LOG YES
LOCATE TABLESPACE tgt dbname.tgt tsname PAGE X'0'
VERIFY OFFSET X'0C' DATA X'src-dbid-src-psid'
REPLACE OFFSET X'0C' DATA X'tgt-dbid-tgt-psid'
DUMP OFFSET X'0C' LENGTH 4
/*
```
For an index space:

```
//REPAIR EXEC DSNUPROC,SYSTEM=tgt_ssid,
// UID='REPR',UTPROC=''
//DSNUPROC.SYSIN DD *
REPAIR OBJECT LOG YES
LOCATE INDEX tgt_creator.tgt_indexspace_name PAGE X'0'
VERIFY OFFSET X'0C' DATA X'src-dbid-src-isobid'
REPLACE OFFSET X'0C' DATA X'tgt-dbid-tgt-isobid'
DUMP OFFSET X'0C' LENGTH 4
/*
```

---

**DB2 object cloning considerations**

Depending on the version of DB2 and type of access required of the target objects, further actions may be required during the cloning process. Additionally, care must be taken when cloning objects that have index names that are greater than eight characters and that may be allocated to the target system at the time of cloning.

**High-used RBA considerations**

When the source DB2 high RBA is higher than the high RBA of the target DB2 instance, the REPAIR utility issues an error when attempting to modify the DBID and PSID:

- The page version number in the database page header is outside the valid range of values (reason code 00C200C1).
- If this happens, the clone can only be used for read-only purposes.

This issue is resolved by resetting the HURBA of the target DB2 to be higher (or equal) to the HURBA of the source DB2. This is achieved by performing a DB2 conditional restart on the target DB2 system specifying a new ending RBA value. This problem does not exist when cloning DB2 objects within the same DB2 instance or across two data sharing instances.
Table 4 summarizes the DB2 access types allowed when cloning data between DB2 instances.

<table>
<thead>
<tr>
<th>RBA Values</th>
<th>Read only</th>
<th>Read/Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source RBA &gt; Target RBA</td>
<td>Read access allowed. Repair level ID required. Repair ISOBID, OBID, and DBID for the index may be required.(^a)</td>
<td>Update not allowed. Repair PSID, DBID on table space not allowed.</td>
</tr>
<tr>
<td>Source RBA &lt; Target RBA</td>
<td>Read access allowed. Repair ISOBID, DBID, PSID for the index may be required.(^a)</td>
<td>Update allowed. Repair DBID, PSID required and allowed.</td>
</tr>
</tbody>
</table>

\(^a\) Refer to “Resetting the level ID on the target” on page 104 to determine if this is required.

\(^b\) If Source RBA > Target RBA and read/write access is required, a DB2 conditional restart can be done to reset the target DB2 tables RBA value to higher than the source DB2 high RBA.

Index considerations

When an index is created with a name longer than eight characters, DB2 assigns an index space name that is eight characters long. Indexes with the same name that are created in a different DB2 instance or in a different database within the same DB2 system could be assigned different index space names. When performing the index cloning, source and target dataset names must be matched.

Buffer pool considerations

If a cloned table space was in use by the target DB2 instance before the cloning, DB2 buffers could contain pages with old data. Consider doing one of the following to avoid errors because of old pages:

- Deallocate the buffer pool by changing VPSIZE to 0, and then allocate it again.
- Alter the table space to use a different buffer pool, and then, if required, change it back.

Note: It is not sufficient to stop and start the table space or to externalize the DB2 buffers in any other way. DB2 might still keep the data in those buffers and use them as necessary.

SNAP performance considerations

The following may improve performance for SNAP operations:
1. Use wild cards where possible.

2. Specify `HOSTCOPYMODE (NONE)` instead of `TOLENQF (Y) ENQWAIT (N)`.

3. Specify `VSAMENQMODE (NONE)`.

Refer to the *EMC TimeFinder/Clone Mainframe Snap Facility Product Guide* for further information.

---

**Cloning method usage considerations**

Cloning can be performed using dataset snap, full volume snap, virtual volume snap, or TimeFinder/Mirror operations. The source objects are unavailable for updates for the duration of the dataset snap or volume snap. The length of time of the snap depends on the number of datasets or volumes that are snapped.

Volume-level techniques are best suited for copying big applications that are populated on a fixed set of volumes. Customers that have large ERP environments (like SAP or PeopleSoft) usually have the entire system on a fixed set of volumes and can have many thousands of datasets. Volume-level processes is usually more efficient for this type of cloning operation.

Dataset snap is best suited for cloning DB2 objects that are sparsely populated across a number of volumes. Customers that are in an SMS-managed environment, or do not have a set of dedicated volumes to contain DB2 data, or both, may be more likely to choose dataset snap for cloning data instead of the volume-level processes.

When datasets are sparsely populated on a number of volumes and it is important to minimize the time the source objects are unavailable for updates, a combination of dataset snap and volume-level operation can be used.

Some cloning implementations require creating multiple instances of the database at the same time. These types of implementations may be suited for dataset snap or volume snap processing because multiple copies of the same data can be performed at the same time.

*Figure 23 on page 109* is provided to assist with selecting the best method for cloning DB2 objects.
Cloning of DB2 Objects and Systems

Figure 23  Object cloning method decision process
DB2 system cloning

DB2 systems can be cloned to the same LPAR or to a different LPAR. These procedures are described in “Cloning the DB2 system within the same LPAR” on page 111 and “Cloning a DB2 system to another LPAR” on page 123.

If the source DB2 is a member of a data sharing group, the procedure for cloning a DB2 instance using TimeFinder creates a non-data sharing target DB2. All data is moved in the Symmetrix system but metadata manipulation on the target DB2 instance is still required. Metadata changes involve the following:

- DB2 table space high-level qualifiers (HLQs)
- VCATs
- STOGROUP definitions
- BSDS changes
- DSNDB07

Figure 24 is a simple example of a cloned DB2 system.

**Figure 24** DB2 system cloning

Examples provided within this chapter use the following nomenclature:

- **DBS** = Source DB2 high-level qualifier
- **DBT** = Target DB2 high-level qualifier
- **DBS1** = Source DB2 instance
- **DBT1** = Target DB2 instance
Cloning the DB2 system within the same LPAR

z/OS must be prepared to support the target DB2 instance. Steps 1 through 4 are required prior to cloning the DB2 instance and need to be performed only once:

1. Define the target DB2 system to z/OS. Refer to the DB2 installation guide for documentation on this process.
2. Define the ICF catalogs.
3. Create the DB2 procedures.
4. Create the target DSNZPARM.

Steps 5 through 11 must be performed each time the cloned environment requires a refresh from the production system:

5. Cloning the datasets.
6. Renaming the cloned datasets.
7. Updating the BSDS.
8. Starting the target DB2 instance.
9. Changing the HLQ of user-defined indexes on the DB2 catalog.
11. Altering user objects to use the new HLQ.

These steps are described in the following sections.

Defining the target DB2 system

Define the target DB2 system and its corresponding IRLM system to z/OS:
Cloning of DB2 Objects and Systems

SUBSYS SUBNAME(DBT1) INITRTN(DSN3INI)
INITPARM('DSN3EP,*DBT,S')
SUBSYS SUBNAME(IDBT)

Defining ICF user catalogs

Define ICF user catalogs for the target DB2 and define aliases for the new system and user datasets:

//DBT0CAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
DEFINE UCAT -
   (NAME(DBT.USER.CATALOG) -
    CYLINDERS(20 5) -
    VOL(DBT001) -
    ICFCATALOG ) -
DATA -
   (CYLINDERS(20 5) ) -
INDEX -
   (CYLINDERS(20 5) ) -
DEFINE ALIAS -
   NAME(DBTL) -
   RELATE(DBT.USER.CATALOG) )
DEFINE ALIAS -
   NAME(DBTD) -
   RELATE(DBT.USER.CATALOG) )

Creating DB2 procedures

Create a target set of DB2 and IRLM procedures in the procedures library. The target xxxxMSTR procedure must be updated to use the new high-level qualifier for the target BSDS. The xxxxMSTR must also be changed to start using the DSNZPARM that is created in “Creating the target DSNZPARM.” The target IRLM procedure must be changed to reflect the target IRLM definition.

Creating the target DSNZPARM

Create a new DSNZPARM member for the target DB2 by applying the following changes to a copy of the source DB2 DSNZPARM:

CATALOG=Target DB2 catalog high-level qualifier
IRLMPROC=Target IRLM procedure name
IRLMSID=Target IRLM system name
ARCPFX1=Target archive high-level qualifier
ARCPFX2=Target archive high-level qualifier
Cloning of DB2 Objects and Systems

Cloning the DB2 system

The detailed steps that follows are an example using TimeFinder/Clone full-volume snap:

1. Bring down the source DB2 instance to stop all work and flush the DB2 buffers. In a data sharing environment, all members of the group must be brought down. The ACTIVATET must not proceed until the last data sharing member has been stopped.

2. Activate a snap on all the volumes that contain all the source DB2 datasets: DB2 catalog and directory, user table spaces, active logs, archive logs, BSDS, ICF catalogs, DBRMLIB.DATA, and RUNLIB.LOAD.

   Use the following JCL as an example:

   ```
   //FVOLSNAP EXEC PGM=EMCSNAP
   //QCOUTPUT DD SYSOUT=* 
   //QCINPUT DD * 
   GLOBAL MAXRC(4), REPLACE(YES), -
   COPYVOLID(NO), DIFFERENTIAL(YES) 
   SNAP VOL(SOURCE(VOL(DBS001)) TARGET(VOL(DBT001)))
   SNAP VOL(SOURCE(VOL(DBS002)) TARGET(VOL(DBT002)))
   SNAP VOL(SOURCE(VOL(DBS003)) TARGET(VOL(DBT003)))
   SNAP VOL(SOURCE(VOL(DBS004)) TARGET(VOL(DBT004)))
   SNAP VOL(SOURCE(VOL(DBS005)) TARGET(VOL(DBT005)))
   SNAP VOL(SOURCE(VOL(DBS006)) TARGET(VOL(DBT006)))
   SNAP VOL(SOURCE(VOL(DBS007)) TARGET(VOL(DBT007)))
   SNAP VOL(SOURCE(VOL(DBS008)) TARGET(VOL(DBT008)))
   ACTIVATE ( CONSISTENT (YES) )
   /*
   3. Bring up the source DB2 instance and resume normal processing.

   The cloned DB2 data now exists on the target volumes but the datasets are not cataloged. To use the cloned datasets in the same z/OS image, the datasets must be renamed and re-cataloged in the target ICF user catalog created in “Defining ICF user catalogs” on page 112.

STORPROC=Target stored-procedure procedure name

If the source DB2 is a member of a data sharing group, change DSHARE to NO in the DSN6GRP macro in the target DSNZPARM.

Assemble and link edit the new DSNZPARM to the DB2 exit library, and give it a different name from the source system DSNZPARM.
Renaming the cloned datasets

The TimeFinder Utility (TFU) completes the dataset cloning process by relabeling the volumes and renaming the datasets. After the renaming process is complete, EMC recommends that you verify that the number of target datasets is equal to the number of source datasets. The number of target datasets may be different from the number of source datasets if you are going from data sharing to non-data sharing. The data sharing instance would have additional logs, BSDS, and sort work datasets with the same high-level qualifier.

1. TFU is used to rename DB2 datasets to use the new target high-level qualifiers for the target DB2 instance.

A sample TFU job to perform the relabel and rename follows:

```plaintext
//EMCTFU EXEC PGM=EMCTFU
//SYSOUT DD SYSOUT=* 
//TFINPUT DD * 

PROCESS CUU=4001, VOLSER=DBT001, BOTH
PROCESS CUU=4002, VOLSER=DBT002, BOTH
PROCESS CUU=4003, VOLSER=DBT003, BOTH
PROCESS CUU=4004, VOLSER=DBT004, BOTH
PROCESS CUU=4005, VOLSER=DBT005, BOTH
PROCESS CUU=4006, VOLSER=DBT006, BOTH
PROCESS CUU=4007, VOLSER=DBT007, BOTH
PROCESS CUU=4008, VOLSER=DBT008, BOTH

CATALOG DBT.USER.CATALOG, DEFAULT
SOURCECATALOG DEFAULT=NO, DIRECT=YES

RENAME DBSD.DSNDBC.DSNDDB01.*, DBTD.DSNDBC.DSNDDB01.*, CATALOG=DBT.USER.CATALOG
RENAME DBSD.DSNDBC.DSNDDB06.*, DBTD.DSNDBC.DSNDDB06.*, CATALOG=DBT.USER.CATALOG
RENAME DBSD.DSNDBC.DSNDDB07.*, DBTD.DSNDBC.DSNDDB07.*, CATALOG=DBT.USER.CATALOG
RENAME DBSD.DSNDBD.DSNDDB01.*, DBTD.DSNDBD.DSNDDB01.*, CATALOG=DBT.USER.CATALOG
RENAME DBSD.DSNDBD.DSNDDB06.*, DBTD.DSNDBD.DSNDDB06.*, CATALOG=DBT.USER.CATALOG
RENAME DBSD.DSNDBD.DSNDDB07.*, DBTD.DSNDBD.DSNDDB07.*, CATALOG=DBT.USER.CATALOG
RENAME DBSL.LOG*, DBTL.LOG*
```
If the source DB2 instance is a member of a data sharing group, then the work file database datasets that were cloned from a data sharing source should be renamed to DSNDB07 to support the non-data sharing target DB2. Figure 25 on page 116 shows the configurations for the source and target DB2 instances, where the source DB2 is a member in a three-way, data sharing environment. Note that the target DB2 instance has a different system ID because it is created within the same LPAR as the source DB2 instance.
Cloning of DB2 Objects and Systems

Figure 25 Cloning—data sharing to non-data sharing on the same LPAR

Updating the BSDS

The next step in the cloning process is to update the target BSDS to reflect the new high-level qualifier for the system datasets. This is done using the DB2 Change Log Inventory utility (DSNJU003). Backing up the BSDS before updating is recommended. In the following example, the high-level qualifier for the target DB2 instance datasets (DBTL) is specified in the NEWCAT statement. The four target active log dataset definitions are deleted and redefined. The start RBA and end RBA for the target active log datasets must be specified. To find these RBA values, run the DB2 utility DSNJU004 on the target BSDS, and then edit the output. Refer to “REXX Example #2” on page 286 for the sample edit macro code to assist with this process.

The following JCL is an example of using the DB2 Change Log Inventory utility to update the BSDS.

```
//DBT0LOG EXEC PGM=DSNJU003
//SYSUT1 DD DISP=OLD,DSN=DBTL.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DBTL.BSDS02
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
NEWCAT VSAMCAT=DBTL
```
Cloning of DB2 Objects and Systems

* Delete the old log names
*DELETE DSNAME=DBSL.LOGCOPY1.DS01
DELETE DSNAME=DBSL.LOGCOPY2.DS01
DELETE DSNAME=DBSL.LOGCOPY1.DS02
DELETE DSNAME=DBSL.LOGCOPY2.DS02
DELETE DSNAME=DBSL.LOGCOPY1.DS03
DELETE DSNAME=DBSL.LOGCOPY2.DS03
DELETE DSNAME=DBSL.LOGCOPY1.DS04
DELETE DSNAME=DBSL.LOGCOPY2.DS04

* Define the new log names
*NEWLOG DSNAME=DBTL.LOGCOPY1.DS03,COPY1,
   STARTRBA=00211133B000,ENDRBA=00211DBAFFF
NEWLOG DSNAME=DBTL.LOGCOPY2.DS03,COPY2,
   STARTRBA=00211133B000,ENDRBA=00211DBAFFF
NEWLOG DSNAME=DBTL.LOGCOPY1.DS04,COPY1,
   STARTRBA=00211DBB000,ENDRBA=00212A83AFF
NEWLOG DSNAME=DBTL.LOGCOPY2.DS04,COPY2,
   STARTRBA=00211DBB000,ENDRBA=00212A83AFF
NEWLOG DSNAME=DBTL.LOGCOPY1.DS01,COPY1,
   STARTRBA=00212A83B000,ENDRBA=0021372BAFFF
NEWLOG DSNAME=DBTL.LOGCOPY2.DS01,COPY2,
   STARTRBA=00212A83B000,ENDRBA=0021372BAFFF
NEWLOG DSNAME=DBTL.LOGCOPY1.DS02,COPY1,
   STARTRBA=0021372BB000,ENDRBA=002143D3AFF
NEWLOG DSNAME=DBTL.LOGCOPY2.DS02,COPY2,
   STARTRBA=0021372BB000,ENDRBA=002143D3AFF

/*

If the source DB2 is a member of a data sharing group, add the
DATASHR DISABLE command to the DSNJU003 job to set data
sharing mode to non-data sharing in the target BSDS.

Starting the target DB2 system

The target DB2 instance can now be started. The target DB2 instance
has no metadata about the user table spaces, indexes, and datasets on
the target volumes. In the target DB2 instance, the DB2 catalog still
points to the source DB2 VCAT. Therefore, even though
the underlying datasets have been cloned and renamed, DB2 must be
made aware of these new dataset names, by changing the DB2
catalog references. To prevent the newly cloned DB2 instance from
allowing access to objects in the original system, start the cloned
system with ACCESS(MAINT).
If cloning from a data sharing environment, disable data sharing for the target DB2 instance. In this case, a cold start of the target DB2 is enforced to prevent outstanding units of work from being backed out. However, in this cloning procedure, the cold start is essentially the same as a normal (warm) restart because all members of the data sharing group were brought down prior to when the TimeFinder copy was created. Stopping all source data sharing members ensures there are no outstanding units of work to back out. Simply reply Y to the WTOR (Write to Operator), indicating that a conditional restart is going to be done.

**Changing the HLQ of user-defined indexes on the DB2 catalog**

Users and independent software vendors sometimes define indexes on the DB2 catalog to support catalog queries. These indexes must be altered to use the new VCAT specified in "Renaming the cloned datasets" on page 114. The following sample JCL can be used to perform an alter VCAT of the user indexes to DBTL.

```plaintext
//DBT0IRU EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//SYSTSIN DD *  
DSN SYSTEM(DBT1)  
-STOP DATABASE(DSNDB06) SPACENAM(USERIX1)  
-STOP DATABASE(DSNDB06) SPACENAM(USERIX2)  
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) -LIBRARY('DBT.RUNLIB.LOAD')  
-START DATABASE(DSNDB06) SPACENAM(USERIX1)  
-START DATABASE(DSNDB06) SPACENAM(USERIX2)  
END  
/*  
//SYSN DD *  
SET CURRENT SQLID='SQLID2';  
ALTER INDEX "CATALOGINDEX-1" USING VCAT DBTL;  
ALTER INDEX "CATALOGINDEX-2" USING VCAT DBTL;  
*/
```

**Changing HLQ of work file database (DSNDB07)**

DSNDB07 is a user-managed database with multiple table spaces defined within it. At this point in the procedure, the underlying VSAM datasets have been renamed, but the DB2 catalog still points to the source DB2 datasets. Correcting this catalog reference requires dropping the work file database and recreating the database and all of its table spaces with the correct VCAT name. When dropping a database that has user-managed table spaces, DB2 does not attempt
Cloning of DB2 Objects and Systems

to delete the underlying VSAM datasets. When user-defined table spaces are recreated, DB2 does not attempt to define the VSAM datasets, and it uses the datasets created by the rename process.

The following example changes the DB2 catalog definition for DSNDDB07 definitions without deleting the underlying datasets:

```sql
//STEP1 EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD *
DSN SYSTEM(DBT1)
-STOP DATABASE(DSNDB07)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) PARM('RC0') -
LIB('DBT.RUNLIB LOAD')
END
/*
//SYSIN DD *
DROP DATABASE DSNDDB07 ;
COMMIT;
CREATE DATABASE DSNDDB07;
/*
/*
//STEP2 EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD *
DSN SYSTEM(DBT1)
-STOP DATABASE(DSNDB07)
RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) -
LIB('DBT.RUNLIB LOAD')
-START DATABASE(DSNDB07)
END
/*
//SYSIN DD *
CREATE TABLESPACE DSN4K01 IN DSNDDB07
BUFFERPOOL BP0
CLOSE NO
USING VCAT DBTD;
CREATE TABLESPACE DSN4K02 IN DSNDDB07
BUFFERPOOL BP0
CLOSE NO
USING VCAT DBTD;
CREATE TABLESPACE DSN4K03 IN DSNDDB07
BUFFERPOOL BP0
CLOSE NO
USING VCAT DBTD;
CREATE TABLESPACE DSN4K04 IN DSNDDB07
BUFFERPOOL BP0
CLOSE NO
USING VCAT DBTD;
CREATE TABLESPACE DSN32K01 IN DSNDDB07
```
BUFFERPOOL BP32K
CLOSE NO
USING VCAT DBTD;
CREATE TABLESPACE DSN32K02 IN DSNDB07
BUFFERPOOL BP32K
CLOSE NO
USING VCAT DBTD;
/*
If the source DB2 instance is a member of a data sharing group, the original data sharing work file databases should be dropped, and DSNDB07 should be created to support the non-data sharing target environment. The remaining work file databases that were cloned from the remaining members are not used for the target DB2 instance and may be dropped, and corresponding datasets may be deleted.

**Altering user objects to use the new HLQ**

The final step in the DB2 instance cloning process is to update the target DB2 catalog to reflect the new high-level qualifier of the cloned DB2 datasets:

1. Create new STOGROUPS with the new volumes and the new VCAT name.
2. Stop all target user databases.
3. Execute an ALTER TABLESPACE and ALTER INDEX statement with the USING STOGROUP parameter, specifying the newly created STOGROUPs.
4. Start all stopped target user databases.
5. If any existing DDL references the old STOGROUP (source STOGROUP definition), the STOGROUP should be dropped at this point and recreated using the new volume definitions and the new VCAT name. If this is not done, objects created by DDL referencing this old STOGROUP are created on the source DB2 volumes with the source DB2 VCAT name.

The following sample DB2 catalog query generates the ALTER statements required for step 3 in the previous procedure. Refer to “REXX Example #3” on page 289 for a sample REXX program that generates ALTER statements with embedded COMMITs.

```sql
SELECT 'ALTER TABLESPACE ' CONCAT STRIP(DBNAME) CONCAT '.
   CONCAT STRIP(TSNAME) CONCAT ' USING STOGROUP NEWSG1;'
FROM SYSIBM.SYSTABLEPART
WHERE PARTITION = 0
```
AND DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')
UNION
SELECT 'ALTER TABLESPACE ' CONCAT STRIP(DBNAME) CONCAT '.
   ' CONCAT STRIP(TSNAME) CONCAT ' PART ' CONCAT DIGITS(PARTITION)
   CONCAT ' USING STOGROUP NEWSG1;' FROM SYSIBM.SYSTABLEPART
   WHERE PARTITION > 0
   AND DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')
UNION
SELECT 'ALTER INDEX ' CONCAT STRIP(B.CREATOR)
   CONCAT '.' CONCAT STRIP(A.IXNAME)
   CONCAT ' PART ' CONCAT DIGITS(PARTITION)
   CONCAT ' USING STOGROUP NEWSG1;' FROM SYSIBM.SYSINDEXPART A,SYSIBM.SYSINDEXES B
   WHERE A.IXNAME=B.NAME
   AND B.DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')
   AND A.PARTITION=0
UNION
SELECT 'ALTER INDEX ' CONCAT STRIP(B.CREATOR)
   CONCAT '.' CONCAT STRIP(A.IXNAME)
   CONCAT ' PART ' CONCAT DIGITS(PARTITION)
   CONCAT ' USING STOGROUP NEWSG1;' FROM SYSIBM.SYSINDEXPART A,SYSIBM.SYSINDEXES B
   WHERE A.IXNAME = B.NAME
   AND B.DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')
   AND A.PARTITION > 0 ;
/

Performance recommendations for the rename process

This section describes actions that can be taken to minimize the cloning execution time:

- **TimeFinder Utility recommendations**
  - Specify an empty UCAT so no cleanup is required, or delete and redefine the UCAT. Specifying a temporary or empty user catalog in the CATALOG statement eliminates the time spent on the cleanup phase.
  - Use SOURCECATALOG DEFAULT=NO and DIRECT=YES: DEFAULT=NO instructs TFU to ignore the source ICF catalog, which is not required for this operation. DIRECT=YES allows TFU to make some ICF catalog changes directly instead of invoking IDCAMS.
  - Direct TFU output to disk rather than the JES output queue.
  - Implement VLF for the user catalog that is specified in the CATALOG statement to be used by TimeFinder Utility processing. Refer to the z/OS: MVS Initialization and Tuning
Reference for details. SYS1.PARMLIB member COFVLFxx should include CLASS NAME IGGCAS and the specified user catalog as an eligible major name (EMAJ).

◆ GRS recommendations

- Change RING to STAR configuration for SYSPLEX. Review the GRS environment. Run time can be shorter in a GRS STAR configuration than in a RING configuration when in a SYSPLEX and there are more than two systems.

- In a RING configuration, check the GRSCNFxx parmlib member for RESMIL and ACCELSYS parameter specifications. Reducing these to smaller values speeds up runtime. Verify that all systems in the GRS RING have the same settings (when the settings do not agree, the highest value is used). The following are values for RESMIL and ACCELSYS that significantly improved performance in a lab environment:
  
  RESMIL (default is 10 - reduce to 1)
  
  ACCELSYS (default is 99 - reduce to 2)

- Uniform distribution of datasets across volumes. If the DB2 environment is located on a relatively small number of volumes, the renaming process may be longer than if the data is spread across a larger number of volumes. Processing a large number of datasets on a small number of volumes restricts TFU parallelism.

◆ Other recommendations

- The cloning process requires that the database datasets of the target DB2 be renamed using the TFU, and that the DB2 catalog entries for these datasets be altered in DB2 to reflect the new names. These two processes can be executed in parallel, provided the target DB2 instance can be started.

- This optimization of the process can be achieved by first renaming the target DB2 system datasets (DB2 catalog, DB2 directory, BSDS, logs), and then restarting the target system with ACCESS(MAINT). The TFU job to rename the remaining datasets and the alter job to update the DB2 catalog entries for these datasets, can then be submitted in parallel, speeding up the overall cloning process.
Cloning a DB2 system to another LPAR

Use the following procedure to clone a DB2 system to an LPAR that does not share ICF catalogs with the source LPAR. If the target DB2 instance resides on an LPAR that shares the same ICF catalogs as the source DB2, then use the process described in “Cloning the DB2 system within the same LPAR” on page 111.

The procedures in this section assume that the target z/OS system has the same I/O system definitions as the source z/OS system. The target z/OS system must be prepared to support a new DB2 instance. Steps 1 through 3 are required prior to cloning the DB2 environment, and they need to be performed only once:

1. Define the target DB2 system.
2. Create DB2 procedures.
3. Create the target DSNZPARM.

Steps 4 through 6 must be performed each time the cloned environment requires a refresh from the production system:

4. Clone the datasets.
5. Associate the source data to the target environment.
6. Start the target DB2 instance.

These steps are described in the following sections.

Defining the target DB2 system

Define the target DB2 system and its corresponding IRLM system to z/OS. These procedures assume that source data sharing groups will be reduced to a single target, non-data sharing DB2.

```bash
SUBSYS SUBNAME(DBT1) INITRTN(DSN3INI)
INITPARM('DSN3EP,*DBT1,S')
SUBSYS SUBNAME(IRT1)
```

The SDSNEXIT, SDSNLOAD, SDXRRESL, and SDSNLINK libraries must be APF-authorized. The hlq.SDSNLINK must be in the linklist.

Creating the DB2 procedures

Replicate the DB2 and IRLM procedures in the procedures library on the target LPAR. These procedures are identical to the procedures used by the source DB2 instance.
Cloning of DB2 Objects and Systems

Creating the target DSNZPARM

If the source DB2 is a member of a data sharing group, change DSHARE to NO in the DSN6GRP macro in the new DSNZPARM. Assemble and linkedit the new DSNZPARM to the DB2 SDSNEXIT library on the target LPAR.

Copying the source data volumes

Perform a TimeFinder array-based copy to replicate the source devices. Shut down the source DB2 instance to achieve a system-wide point of consistency just prior the SPLIT/ACTIVATE. The detailed steps are as follows:

1. Use TimeFinder to copy all the volumes that contain all the source DB2 datasets: DB2 catalog and DB2 directory, user table spaces, active logs, BSDS, ICF catalogs, DBRMLIB.DATA, and RUNLIB.LOAD.

2. Bring down the source DB2 to stop all work and flush the buffers. In a data sharing environment, all members of the group must be brought down. The SPLIT/ACTIVATE must not proceed until the last data sharing member has been stopped.

3. SPLIT/ACTIVATE the TimeFinder copy.

4. Bring up the source DB2 subsystem and resume normal processing on it.

The replicated DB2 datasets now exist on the target volumes, but the datasets are not cataloged.

Associating source data to the target environment

The ICF user catalog must be imported from the source LPAR into the target LPAR and aliases must be created for the high-level qualifier (tables). The ICF user catalogs should be replicated with the source volumes so they can be connected to the master catalog. Steps to accomplish these tasks are as follows:

1. Vary the target volumes online to the target z/OS.

2. Import connect the ICF user catalogs:

   ```bash
   // IMPORT EXEC PGM=IDCAMS
   // SYSPRINT DD SYSOUT=*
   // SYSSIN DD *
   IMPORT -
   OBJECTS ( -
    (DBT.USER.CATALOG -
    VOLUME (DBT000) -
   ```
DEVICETYPE(3390)) –
CONNECT CAT(CATALOG.MASTER)
/*

3. Define all aliases:

//DBT0CAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
DEFINE ALIAS –
   ( NAME(DBT1) –
     RELATE(DBT.USER.CATALOG) )
DEFINE ALIAS –
   ( NAME(DBTD) –
     RELATE(DBT.USER.CATALOG) )
DEFINE ALIAS –
   ( NAME(DBTL) –
     RELATE(DBT.USER.CATALOG) )
/*

Starting the target DB2 instance

If the source DB2 is a member of a data sharing group, execute the DATASHR DISABLE command using the DSNJU003 utility to set data sharing mode to non-data sharing. Data sharing must be disabled for the target DB2 instance. Figure 26 on page 126 shows the configurations for the source and target DB2 instances, where the source DB2 is a member of a three-way data sharing environment. In Figure 26, DB2A is a member in the source data sharing group and has been replicated onto a different LPAR, but the DB2 system ID remains the same.
The change from data sharing to non-data sharing for the target DB2 instance enforces a cold start of the target DB2 instance to prevent outstanding units of work from being backed out. However, in this cloning procedure, the cold start is essentially the same as a normal (warm) restart because bringing down all members of the data sharing group prior to the TimeFinder copy being made ensures there are no outstanding units of work to back out. Simply reply Y to the WTOR, indicating that a conditional restart will be done.

Work database considerations for data sharing

If the source DB2 is a member of a data sharing group, the original work file databases datasets of the data sharing members must be renamed to DSNDB07. This could be done using the IDCAMS ALTER command.

DSNDB07 is a user-managed database with multiple table spaces defined within it. At this point in the procedure, the underlying VSAM datasets are renamed but the DB2 catalog still points to the source DB2 datasets. Correcting this DB2 catalog reference requires dropping the work file database, and recreating the database and all table spaces with the correct database name. When dropping a
database that has user-managed table spaces, DB2 does not attempt to delete the underlying VSAM datasets. When the user-defined table spaces are recreated, DB2 does not attempt to define the VSAM datasets, and uses the datasets created by the rename process.

The following example changes the DB2 catalog definition without deleting the underlying datasets:

```sql
//STEP1 EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD * 
  DSN SYSTEM(DBT1) 
  -STOP DATABASE(data sharing work database) 
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) PARM('RC0') - 
      LIB('DBT.RUNLIB.LOAD') 
  END 
  / * 
//SYSEX DD * 
  DROP DATABASE data sharing work database ; 
  COMMIT; 
  CREATE DATABASE DSNDB07 ; 
  / * 
  / * 
//STEP2 EXEC PGM=IKJEFT01, 
//SYSTSPRT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSTSIN DD * 
  DSN SYSTEM(DBT1) 
  -STOP DATABASE(DSNDB07) 
  RUN PROGRAM(DSNTIAD) PLAN(DSNTIA91) - 
      LIB('DBT.RUNLIB.LOAD') 
  -START DATABASE(DSNDB07) 
  END 
  / * 
//SYSEX DD * 
  CREATE TABLESPACE DSN4K01 IN DSNDB07 
      BUFFERPOOL BP0 
      CLOSE NO 
      USING VCAT DBTD; 
  CREATE TABLESPACE DSN4K02 IN DSNDB07 
      BUFFERPOOL BP0 
      CLOSE NO 
      USING VCAT DBTD; 
  CREATE TABLESPACE DSN4K03 IN DSNDB07 
      BUFFERPOOL BP0 
      CLOSE NO 
      USING VCAT DBTD; 
  CREATE TABLESPACE DSN4K04 IN DSNDB07 
      BUFFERPOOL BP0 
      CLOSE NO 
      USING VCAT DBTD; 
```
CREATE TABLESPACE DSN32K01 IN DSNDDB07
    BUFFERPOOL BP32K
    CLOSE NO
    USING VCAT DBTD;
CREATE TABLESPACE DSN32K02 IN DSNDDB07
    BUFFERPOOL BP32K
    CLOSE NO
    USING VCAT DBTD;

/*

Recoverability of the cloned target DB2 system

When cloning from a data sharing environment to a non-data sharing environment, archive logs and active logs created prior to the cold start cannot be used for recovery. This is because multiple logs that exist in a data sharing environment cannot be merged into a single log for a non-data sharing environment. When cloning a data sharing DB2 system, within or across LPARs, the cloned DB2 system is restartable, but not recoverable.

When cloning non-data sharing environments, the cloned DSNZPARM contains the prefix of past and future archive logs. Cloning archive log datasets within an LPAR requires using a different archive log prefix to prevent duplicate archive log dataset names. Cloning and renaming an archive log requires that the archive log be on DASD. It may be unreasonable to maintain more than a few archive logs on disk. Therefore, it may not be practical to clone a DB2 system and ensure recoverability within the same LPAR.

If recoverability is required, then image copies can be taken on the cloned target DB2 system. This establishes a starting point for recoverability.
This chapter describes the ways that EMC Symmetrix subsystems can enhance a DB2 backup strategy using local array-based replication techniques.

- Overview ................................................................. 130
- Recoverable vs. restartable copies of databases ............... 131
- Database organization to facilitate recovery .................. 132
- EMC replication for DB2 system backup ........................ 135
- Creating a remote backup .......................................... 144
- Backup method considerations .................................... 147
- Creating multiple array copies ..................................... 147
- Reporting on dataset placement on backup volumes ......... 149
- DB2 image copy considerations .................................... 150
- Considerations for 32 KB pages .................................. 150
- SYSLGRNX and SYSCOPY recovery .............................. 151
- Operations with LOG NO considerations ....................... 151
- Backing up DB2 objects with dataset snap ..................... 151
- Backup method comparison ........................................ 154
- Symmetrix arrays with DB2 system point-in-time recovery .... 155
- Backing up DB2 V9 for object-level recovery .................. 158
- DB2 V10 FlashCopy Image Copy .................................... 159
- Space Efficient FlashCopy .......................................... 160
- Rocket Backup and Recovery for DB2 on z/OS ................. 161
Overview

The size and complexity of information systems have vastly increased in recent years. The requirement for high availability is more important than ever. The 24/7 availability requirement for some databases makes it extremely difficult to perform database maintenance operations and to prepare for fast recovery in case of a disaster. Acceptable backup solutions are ones that do not compromise availability or performance.

For large DB2 systems, an array-based replication solution is recommended to solve the performance and availability issues. This solution offers a quick way of creating copies of data (instead of using DB2 utilities) and has no effect on production availability and performance. The array-based copy can be the source of an out-of-band backup and can provide very fast recovery. It simplifies the recovery process since the same procedures are used for all objects in the environment.

This chapter describes the ways to create array-based copies of DB2 systems using the EMC TimeFinder family of products:

- TimeFinder/Mirror
- TimeFinder/Clone Full-Volume Snap
- TimeFinder/Snap
- TimeFinder/Consistency Groups

The procedures for recovering from this kind of backup are discussed in Chapter 6, “DB2 Recovery Procedures”.

Before covering these capabilities, it is necessary to review some terminology and best practices for DB2 for z/OS database layouts that can facilitate and enhance the backup and restore process.
Recoverable vs. restartable copies of databases

The array-based replication technologies for backup described in this section can create two types of database copies: Recoverable or restartable. The differences between these two types of database copies can be confusing. A clear understanding of the differences between the two is critical to ensure that the recovery goals for a database can be met.

Recoverable disk copies

A recoverable database copy is one to which logs can be applied and the database rolled forward to a point in time after the copy was created. A recoverable DB2 system copy is intuitively easy for DBAs to understand since maintaining recoverable copies, in the form of backups, is an important DBA function. In the event of a failure, the ability to recover the system not only to the point in time when the last backup was taken, but also to roll forward subsequent transactions up to the point of failure, is a key feature of the DB2 system.

Creating recoverable images of DB2 for z/OS databases with EMC replication technology requires that the database be shut down when it is copied, or if a running database is to be replicated, the database must be in log-suspend mode, unless RBR (Rocket Backup and Recovery for DB2 on z/OS) is being used. RBR is described later in this chapter.

Restartable disk copies

If a copy of a running DB2 system is created using EMC Consistency technology without putting the database in log-suspend mode, the copy is a DBMS restartable image. This means that when the replicated DB2 system is started on the copy, it performs its normal restart process to resolve in-flight transactions. DB2 uses the active logs and sometimes the archive logs to return the system to a point of transactional consistency, by either backing out in-flight transactions or completing committed transactions.

Forward recovery using archive logs to a point in time after the disk copy is created is supported on a restartable database copy if the copy was created by RBR, as described later in this chapter.
Database organization to facilitate recovery

When using array-based replication, it is prudent to organize the DB2 datasets in such a way as to facilitate recovery. Since array replication techniques copy volumes at the physical disk level (as seen by the host), all table spaces for a system should be created on a set of disks dedicated to the system and should not be shared with other applications or DB2 systems. Sharing DASD volumes with other applications causes unnecessary work for the array and wasted space on the target volumes.

In addition to isolating the database to be copied onto its own dedicated volumes, the database should also be physically divided into two parts: The recovery structures and the data structures. The volumes for the recovery structures must contain:

- Bootstrap dataset (BSDS)
- Active logs
- Archive logs optionally (if on disk)
- Dedicated ICF user catalog for the preceding objects

The database volumes for the data structures must contain:

- DB2 catalog
- DB2 directory
- Table spaces and index spaces
- Dedicated ICF user catalog for the preceding objects

This physical segregation allows the two parts to be manipulated independently if a recovery becomes necessary. The requirement for ICF catalogs to reside on the volumes that contain the datasets referenced by the catalog is due to the anticipated use of full-volume restore. In the case of a full-volume restore, it is necessary to restore all the volumes that contain data referenced by the catalog. For this reason it is advisable not to catalog into these user catalogs anything that is not a dataset required by DB2.

If full-volume restore is not required (that is, only dataset restore is required) the preceding ICF catalog placement is not necessary.

Figure 27 on page 133 depicts a TimeFinder setup where the data structures and recovery structures have been separated onto their own volumes.
Figure 27  **Recommended dataset placement**

**Note:** This figure is used as the basis for the replication procedures discussed in the following sections.

---

**Preparing for an array-based backup**

Several DB2 catalog and directory table spaces require special attention to ensure that they are recoverable from a full-volume backup. The following table spaces, along with their indexes, do not have entries in `DSNDB01.SYSLGRNX`, even when defined with `COPY YES`:

- `DSNDB01.SYSUTILX`
- `DSNDB01.DBD01`
- `DSNDB01.SYSLGRNX`
- `DSNDB01.SCT02`
- `DSNDB01.SPT01`
- `DSNDB06.SYSCOPY`
- `DSNDB06.SYSGROUP`
During a `RECOVER LOGONLY` of these objects, the log range is scanned for updates to be applied, ranging from the HPGRBRBA of the table space (kept in the header page of each table space), to the end of the log. Theoretically, if a table space from this list has not been updated for a long time, then the HPGRBRBA contains an old RBA value, and corresponding archive logs might have already been deleted. Under these circumstances, recovery might fail for this table space.

To prevent such a failure from happening, a `DB2 QUIESCE WRITE(YES)` should be executed for the seven listed table spaces prior to creating a full-volume backup. This updates the HPGRBRBA with the current RBA, ensures recoverability of these seven table spaces, and shortens the recovery process. The HPGRBRBA for indexes that may exist for these table spaces, if they are defined `COPY YES`, are also updated at this time. Indexes that are not defined `COPY YES` do not have the HPGRBRBA advanced when the `DB2 QUIESCE WRITE(YES)` is issued for the table space. If an `ALTER` is used to change the copy value on an index, then it should be stopped and restarted to allow the HPGRBRBA to be updated. To quiesce these table spaces, use the following sample JCL:

```bash
//UT1 EXEC
DSNUPROC=SYSTEM=DSN1,UID='QUIESCE1',UTPROC=''
  LIB='DSN810.PROD.SDSNLOAD'
//DSNUPROC.SYSIN DD *
QUIESCE TABLESPACE DSNDB01.DBD01 WRITE(YES)
QUIESCE TABLESPACE DSNDB01.SYSUTILX WRITE(YES)
QUIESCE TABLESPACE DSNDB01.SYSLGRNX WRITE(YES)
QUIESCE TABLESPACE DSNDB01.SPT01 WRITE(YES)
QUIESCE TABLESPACE DSNDB01.SCT02 WRITE(YES)
QUIESCE TABLESPACE DSNDB06.SYSGROUP WRITE(YES)
QUIESCE TABLESPACE DSNDB06.SYSCOPY WRITE(YES)
/*
EMC replication for DB2 system backup

EMC volume-based replication can be used to make a copy of a DB2 system, and this copy can be used as a source for backing up the DB2 system to tape. A database backup process using volume-replication technology typically includes some or all of the following steps, depending on the copying mechanism selected and the desired usage of the system backup:

1. Preparing the array for replication.
2. Conditioning the source system.
3. Making a copy of the DB2 volumes.
4. Resetting the source system.
5. Presenting the target volumes to the host.
6. Backing up the target volumes.

In all cases but one, operating system capabilities are used to back up the copies of the system volumes in step 6. In other words, the DB2 image copy utility is not used, and no entries are made for the backup in SYSIBM.SYSCOPY.

The exception to this rule is if RBR is used to back up the system. RBR has the capability to copy individual files, or sets of files, from the target volumes to tape and register those files as image copies in SYSIBM.SYSCOPY.

The following six sections describe each of the six steps in detail and whether or not they are needed in a given replication architecture. All the six steps are based on the configuration depicted in Figure 27 on page 133. The source database devices are on UCBs 1000-1007. The target data devices are on UCBs 4000-4007.

Preparing the array for replication

There are only two situations when the array needs to be prepared prior to making the disk copy of the DB2 system:

- When using TimeFinder/Mirror
- When using TimeFinder/Clone Mainframe Snap Facility with pre-copy.

These two situations are described below.
TimeFinder/Mirror preparation

TimeFinder/Mirror requires that the target volumes be synchronized with the source volumes prior to executing the step to create the copy of the system. The synchronization is accomplished using the ESTABLISH function. If this is a second or subsequent time, the RE-ESTABLISH function can also be used.

The establish function can be initiated using the following JCL:

```
//TFESTAB EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD * 
GLOBAL MAXRC=4,WAIT,FASTEST(Y)
ESTABLISH 01,4000-4007,1000-1007 
/*
```

This JCL initiates the copying of the data from the source devices (1000-1007) to the target BCV devices (4000-4007) using Symmetrix resources. The ESTABLISH command can only work with BCV devices so that the target volumes must be defined in the Symmetrix controller as BCV volumes. The source volumes that make up the entire DB2 environment (user datasets, DB2 catalog, DB2 directory, active logs, archive logs, BSDS and the related ICF user catalogs) are copied using this process.

TimeFinder/Clone Mainframe Snap Facility with pre-copy

When TimeFinder/Clone full-volume snap is used there is an option to start copying the tracks from the source to the target in advance of when the DB2 copy is needed. This mode is called pre-copy. It is analogous to the TimeFinder/Mirror ESTABLISH operation except that the target can be activated at any time and does not have to be fully synchronized.

The use of the pre-copy functionality mitigates potential performance impact of redirected reads from the target volumes. In environments with heavily used source volumes, this might be desirable.

In order to use the pre-copy functionality, group processing must be used. The following JCL defines the group DBT1:

```
//EMCTF EXEC PGM=EMCSNAP,REGION=0M
//STEPLIB DD DSN=SYSx.SSNP580.LINKLIB,DISP=SHR
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=* 
//QCOUTPUT DD SYSOUT=* 
//QCINPUT DD * 
DEFINE GROUP DBT1 ( -
    FORCE(YES) -
```
REPLACE(YES) –

GLOBAL PRECOPY(YES),DIFFERENTIAL(YES),MESSAGES(DISPLAY)
SNAP VOLUME (SOURCE( UNIT(1000-1007))
TARGET(UNIT(4000-4007)))
END GROUP
/*

The group definition is stored in the PDS assigned to the EMCGROUP
DDNAME. Following this definition, only the group name and the PDS
are needed to reference the volume pairs. All volumes in the group
are acted on as a set.

Since the group was defined with PRECOPY(YES), all that is needed
now is to start the copying. The JCL to do this is:

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=*  
//QCOUTPUT DD SYSOUT=*  
//QCINPUT DD *
SNAP VOLUME ( GROUP (DBT1) )
/*
```

After this JCL is executed, tracks on the source are copied to the
target. If a differential relationship already exists, only the changed
tracks since the last full-volume snap are copied. If a track is copied in
this phase but is subsequently changed by the host, it is marked as
needing copying again in the next phase.

**Conditioning the source system**

It is always recommended to perform the actions described in
“Preparing for an array-based backup” on page 133. In addition there
are only two ways to create a consistent system copy while DB2 is
actively running. The first is by using Enginuity Consistency Assist
(ECA), which requires no action to “condition” the source system.
The second method is by suspending the log activity while the
system is being copied. The log is suspended using the command:

```
-SET LOG SUSPEND
```

When this command is executed, DB2 takes the following actions:

1. Externalizes log buffers.
2. Takes a system checkpoint (non data-sharing).
3. Updates the BSDS with the high written RBA.
4. Suspends the update activity.
5. Quiesces the writes for 32KB pages and the dataset extensions for all page sizes.

This command affects transaction response time if the logs are not resumed quickly.

The advantage of using the `-SET LOG SUSPEND` command is that the image created while the system is in this state is a recoverable image. The image created when using ECA is a restartable image.

In a data sharing environment, the `-SET LOG SUSPEND` command must be issued to all members of the group.

---

**Making a copy of the DB2 volumes**

This is the point in the process where the copy of the DB2 system is actually created and made available to the host. In most cases, this only requires seconds to complete. Whether or not the DB2 logs are suspended at this time, it is recommended to use ECA to create the disk copy. This has the advantage of provided restartable copy of DB2 and, if the logs are suspended, a recoverable one too. If ECA is not available (that is to say TimeFinder/CG is not licensed), the logs must be suspended before this step. The command syntax using ECA and without ECA is provided in the following examples.

If RBR is used to create the system copy, ECA can be used and provides both recoverable and restartable DB2 system copies without suspending the log.

Each of the three TimeFinder products requires different syntax in this phase. The follow three sections describe how each copy is created.

**TimeFinder/Mirror Split**

The TimeFinder/Mirror `SPLIT` process separates the BCVs from the source devices and makes them available for processing by the host. It is necessary that the BCVs are completely synchronized with the source volumes in order for this command to work. The following JCL performs a consistent split (using ECA):

```
//TFSPLIT EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD *  
GLOBAL MAXRC=4, WAIT, FASTEST(Y)  
SPLIT 01,4000-4007, CONS(GLOBAL(YES))  
/*
```
### TimeFinder/Clone full-volume activate

Unlike TimeFinder/Mirror, TimeFinder/Clone Full-Volume Snap does not require any copying to be performed in advance of the ACTIVATE. So whether or not the pre-copy mechanism was used in step 1, the process is the same for the ACTIVATE.

The target volumes for full-volume snap can be any volumes in the array, including BCVs, as long as the volumes are the same size and same emulation (CKD in this case).

The following JCL can be used to activate the full-volume snap. This makes the copy available immediately after the ACTIVATE has completed. The SYSIN for this JCL shows that ECA is being used because of the presence of the CONSISTENT(YES) keyword. This should be the default even if the DB2 logs have been suspended. If TimeFinder/CG is not licensed, simply remove the CONSISTENT(YES) keyword.

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=*  
//QCOUTPUT DD SYSOUT=*  
//QCINPUT DD *
ACTIVATE ( GROUP (DBT1) CONSISTENT(YES) MESSAGES(DISPLAY) )
/*

After the activation, the target volumes are immediately available to z/OS for processing, even if the copy process has not yet completed. The copy process continues in the background until the point-in-time image at the time of the activation has been copied to the target volumes.

It should be noted that if target volumes are read before all the tracks are copied across, then some of that read activity might be redirected to the source volumes. This may be problematic based on the level of utilization of the source volumes. Using PRECOPY(YES) and DIFFERENTIAL(YES) can mitigate much of this impact.

### TimeFinder/Snap activate

TimeFinder/Snap allows for the use of virtual devices (VDEVs) as targets for a copying operation. Each VDEV presented to an LPAR looks exactly like a real device but does not occupy space within the storage array. Space is consumed only when the VDEV is written to, or when the source device is written to, after the virtual snap is activated.
The following JCL can be used to activate the snap to virtual devices. This makes the copy available immediately after the activate has completed. The input for this JCL shows that ECA is being used because of the presence of the CONSISTENT(YES) keyword on the ACTIVATE command. This should be the default, even if the DB2 logs have been suspended. If TimeFinder/CG is not licensed, simply remove the CONSISTENT(YES) keyword. The target devices 4000-4007 are assumed in this context to be VDEVs.

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=* 
//QCOUTPUT DD SYSOUT=* 
//QCINPUT DD * 
SNAP VOLUME -
   ( 
      SOURCE (UNIT(1000-1007)) - 
      VDEVICE(UNIT(4000-4007)) - 
   ) - 
ACTIVATE (CONSISTENT(YES)) 
/*
```

As soon as the ACTIVATE completes, the copy on the target VDEVs is immediately available for processing. Note that with this replication solution, no data is copied from the source unless the source data is changed. When the data is changed on the source, the Symmetrix controller preserves the point-in-time image on the virtual devices by copying the pre-update data on the source track to a save area and changing the VDEV pointer to point to that track rather than the original track on the source volume. In this manner, the point-in-time data state on the virtual devices is preserved.

If the source data changes a lot over the lifetime of the virtual snap, all the changed data has to be stored in the save area. For this reason, virtual snap is best for data with a low change rate or snaps that are only needed for a short while and then are terminated.

---

**Resetting the source system**

In this part of the process the DB2 system is returned to its normal operating environment. This step is needed only if the -set log suspend command was used earlier. The following command is used to resume logging:

```
-SET LOG RESUME
```
This command puts DB2 back into normal operating mode. In a data sharing environment the log needs to be resumed on every member in the group.

---

**Presenting the target volumes to a host LPAR**

Once the DB2 volumes have been copied, they can be presented to an LPAR for backup. It may seem obvious but it is worth stating here: The target volumes need to have a UCB address assigned in order for the host to be able to read them.

In most cases, the volume needs to be varied online to be accessed by the host LPAR. The exception is if FDR Instant® Backup is being used, which can process the volumes while they are offline.

If a volume needs to be varied online for backup, particular care must be taken to ensure the volume does not contain a duplicate volume serial number. The are several ways this can be managed:

1. Do not copy the volume serial number of the source volume to the target volume. This is available with TimeFinder/Clone and TimeFinder/Snap only. This is accomplished by using the COPYVOLID(NO) parameter on the full-volume snap. Alternatively, you can also use the NEWVOLID(volser) syntax to provide a new volume serial number for the target volume. If the snap volume pre-copy capability is used then the VOLSER of the source volume is copied to the target volume and cannot be changed using the snap process.

2. Use the SPLIT function for TimeFinder/Mirror to change the volume serial during the SPLIT process. This can be accomplished by adding the VOLID syntax to the SPLIT command, as in the following example:

   ```
   SPLIT 1,4000,VOLID(newvol)
   ```

3. Vary the volume online to an LPAR that has no visibility to the source volume. In this way, there is no conflict for the copied volume serial number.

4. Use the TimeFinder Utility to relabel the volume serial on the target volume. The following sample JCL shows how this can be done:

   ```
   //EMCTFU EXEC PGM=EMCTFU
   //SYSOUT DD SYSOUT=*  
   //TFINPUT DD * 
   ```
**Backing Up DB2 Environments**

RELABEL CUU=4000, OLD-VOLSER=DBS001, NEW-VOLSER=DBT001
RELABEL CUU=4001, OLD-VOLSER=DBS002, NEW-VOLSER=DBT002
*/

Except for the technique used in step 3 above, the target volume is only suitable for a full-volume backup because the new volume serial number is different from that which is in the VTOC index dataset name and also in the VVDS dataset name (if the volume contains VSAM datasets).

If step 3 is used, dataset backup is immediately possible by connecting the user catalog on the replicated volumes to the master catalog in the other LPAR.

---

**Backing up the target volumes**

Copying the target volumes to sequential media (typically tape) is the final phase in the backup process. It is rare that organizations can keep enough disk copies on-line to meet all their recovery requirements, so this step is usually necessary. It is particularly important in the case of virtual device copies, as these exist only when the source volumes exist and are not independent copies. Thus, if the source is lost, the virtual device copies are also lost.

The most common backup processes are:

- Full-volume backup with DFDSS
- Full-volume backup with FDR
- Dataset backup

---

**Full-volume backup using DFDSS**

To perform a full-volume DFDSS dump on two target volumes DBT000 and DBT001, use the following sample JCL:

```
//DUMPA EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//DISK1 DD UNIT=3390,DISP=SHR,VOL=SER=DBT000  
//TAPE1 DD DSN=DB2BKP.FULLBKUP.DBT000,  
// UNIT=TAPE,DISP=(NEW,KEEP),LABEL=(1,SL),  
// VOL=SER=TAPE00
//DISK2 DD UNIT=3390,DISP=SHR,VOL=SER=DBT001  
//TAPE2 DD DSN=DB2BKP.FULLBKUP.DBT001,  
// UNIT=TAPE,DISP=(NEW,KEEP),LABEL=(1,SL),  
// VOL=SER=TAPE01
//SYSIN DD *  
DUMP FULL INDD(DISK1) OUTDD(TAPE1) OPT(4)
```
Full-volume backup using FDR

To perform a full-volume FDR dump, using the following sample JCL:

```sql
//DUMP EXEC PGM=FDRDSF
//SYSPRINT DD SYSOUT=*  
//DISK1 DD UNIT=3390, VOL=SER=DBT000, DISP=OLD  
//TAPE1 DD DSN=DB2BKP.FULLBKUP.DBT000,  
//              UNIT=TAPE, DISP=(NEW,KEEP), LABEL=(1,SL),  
//              VOL=SER=TAPE00  
//DISK2 DD UNIT=3390, VOL=SER=DBT001, DISP=OLD  
//TAPE2 DD DSN=DB2BKP.FULLBKUP.DBT001,  
//           UNIT=TAPE, DISP=(NEW,KEEP), LABEL=(1,SL),  
//           VOL=SER=TAPE01  
//SYSIN DD *  
DUMP TYPE=FDR  
/*
```

Dataset backup from the target volume

The backup process can be structured around a dataset dump of the target volumes. This is called a logical backup (as opposed to a physical backup). If a logical dump is taken from the same host, or a different host with shared ICF catalogs, then all datasets must be renamed using the TimeFinder Utility. This rename process can be time consuming if there are many datasets and recovery from it takes longer than from a full-volume dump. If the logical dump is performed from an LPAR that does not share ICF user catalogs with the source LPAR, renaming the datasets to be dumped is not a requirement.

The recovery from a logical backup is more complex than a simple volume-based restore and recovery of the system. For this reason it is not recommended.
Creating a remote backup

In remotely mirrored environments using SRDF/S or SRDF/A a remote consistent copy can be created at the remote location. TimeFinder Consistency Groups, TimeFinder/Mirror or TimeFinder/Clone replication software, and SRDF can all work together to create a restartable copy of a DB2 configuration on a remote Symmetrix subsystem.

The general principles of data organization for DB2, as outlined in “Database organization to facilitate recovery” on page 132, should still be followed but are not as critical for the remote backup. This is because the remote backup is primarily used for restarting the DB2 system and not recovering it.

Figure 28 on page 144 is a high-level view of a remote replication setup using SRDF/S or SRDF/A with target volumes for the TimeFinder copy in the remote Symmetrix subsystem.

To create a remote backup on an R2 Symmetrix array using remote TimeFinder commands and TimeFinder Consistency Groups, use the following procedure and refer to Figure 28. The remote copy can be created using a TimeFinder/Mirror, TimeFinder/Clone, or TimeFinder/Snap operation on the R2s when using SRDF/S. If SRDF/A is being used, only TimeFinder/Mirror and TimeFinder/Clone are supported to make copies off the R2. The following three examples show how to create the remote copy in-band by copying the whole DB2 environment (user datasets, DB2 catalog, DB2 directory, active logs, archive logs, BSDS, and the related ICF catalogs):
1. For TimeFinder/Mirror, reestablish (establish if the first time) the remote BCVs to the R2 devices, followed by a consistent SPLIT. The following JCL can be used to perform a remote ESTABLISH and SPLIT.

```
//RUNUTIL EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=*  
//SYSIN DD *
GLOBAL MAXRC=4,WAIT,FASTEST(Y),CONSISTENT(YES)
ESTABLISH 01,RMT(400,00A0-00A7,01A0-01A7,04)
SPLIT 02,RMT(4000,01A0-01A7,04)
/*
```

In the SYSIN, 00A0-00A7 are the Symmetrix device numbers of the remote BCV devices. The remote R2 devices are 01A0-01A7. The RA group for the SRDF link is 04.

2. For TimeFinder/Clone perform a full-volume SNAP from the R2s to the remote targets.

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=*  
//QCOUTPUT DD SYSOUT=*  
//QCINPUT DD *
GLOBAL CONSISTENT(YES),DIFFERENTIAL(YES)
SNAP VOLUME
(  
   SOURCE( SYMDV#(00A0-00A7))  
   TARGET( SYMDV#(01A0-01A7))  
   REMOTE( UNIT(4000) RAGROUP(04))  
)
ACTIVATE
/*
```

In the JCL, 00A0-00A7 are the Symmetrix device numbers of the remote R2 volumes. The remote targets are 01A0-01A7. The RA group for the SRDF link is 04.

3. For TimeFinder/Snap perform a full virtual device SNAP from the R2s to the virtual device remote targets.

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=*  
//QCOUTPUT DD SYSOUT=*  
//QCINPUT DD *
GLOBAL  
```

Note: With Enginuity 5875 and later, it is possible to create a remote backup using virtual devices and SRDF/A R2 volumes.
GLOBAL CONSISTENT(YES), DIFFERENTIAL(YES)
SNAP VOLUME
   ( -
      SOURCE(SYMDV#(00A0-00A7)) -
      VDEV(SYMDV#(02A0-02A7)) -
      REMOTE(UNIT(4000) RAGROUP(04)) -
   )
ACTIVATE
/*

In the JCL, 00A0-00A7 are the Symmetrix device numbers of the remote R2 volumes. The remote virtual devices are 02A0-02A7. The RA group for the SRDF link is 04.
Backup method considerations

When choosing a backup method using array-based technology there are some specific considerations that should be understood in order to determine the optimal solution. The following is a list of considerations that can impact the exact method chosen:

- The DB2 suspend log and resume log activities can have an impact on transactions. Transactions are held for the duration the log is suspended. Ideally, the log must be suspended for a minimal amount of time. In addition, suspend log can be problematic in a data sharing environment.

- The consistent SPLIT/ACTIVATE has no performance or transactional impact. It requires TimeFinder/Consistency Group in order to be able to utilize ECA (Enginuity Consistency Assist) on the Symmetrix array. Consistent SPLIT/ACTIVATE without suspend log yields a system backup that can be used for system restart and object restore only, unless controlled by RBR.

- TimeFinder/Clone and TimeFinder/Snap can have a COFW (copy on first write) impact on Enginuity levels prior to 5772. For Enginuity 5772 and later, the management of the impact is asynchronous to the host, and the impact is negligible.

- TimeFinder/Clone and TimeFinder/Snap can redirect workload to the source volumes when reading from the targets if the track is indirected. This can be avoided with TimeFinder/Clone by using the pre-copy function or waiting until all the data is copied to the target.

Creating multiple array copies

TimeFinder allows up to eight copies of a source volume while still retaining the incremental relationship between the source and the target. While it would be rare to use all eight copies, it is can be seen that two copies of the data on disk has some distinct advantages:

- If only one set of volume targets are used, and synchronized on a daily basis, there is no volume-based restore point available for the period of time of the resynchronization. Having a second set of volumes supporting the restore means that volume-based restore is always be possible.
Most recoveries are caused by user or application errors. It may take a while to recognize that the data is corrupted, and the logical error might have already been propagated to the volume-based replica of the environment. A second set of backup volumes from a prior point in time may prevent the need for a full-volume restore from tape.

Having multiple volume targets means that the additional targets can be used for other purposes, such as for test, development, and QA.

If practical, EMC recommends having at least two sets of target volumes and to toggle between the two sets when creating DB2 system backup environments. To eliminate the resynchronization exposure, multiple target sets increase the chance of having a good copy on a target volume in case of an event requiring a restore. Restoring the environment from a set of disk volumes is significantly faster than restoring from tape. Figure 29 shows a DB2 environment configured with three sets of target volumes.

![DB2 configured with three sets of target volumes](ICO-IMG-000722)
Reporting on dataset placement on backup volumes

The TimeFinder Utility can be run immediately on the target volumes after a backup is completed to create a recovery report. The RENAME command to a dummy dataset is used to create the recovery report. The rename function processes all volumes in the target set and create a report that contains the mapping of all datasets to their volumes. Information from the recovery report and the tape management system can be used to locate a single dataset for a logical restore, even after a backup to tape has been done. This report should be accessible in the event that a recovery is required.

The following sample JCL can be used to produce this report:

```
//EMCTFU EXEC PGM=EMCTFU
//SYSOUT DD SYSOUT=* 
//TFINPUT DD * 
PROCESS CUU=4120,VOLSER=DSNS00,BOTH 
PROCESS CUU=4121,VOLSER=DSNS01,BOTH *
CATALOG DBT.USER.CATALOG,DEFAULT 
SOURCECATALOG DEFAULT=NO,DIRECT=YES 
RENAME DUMMY.DATA.SET1.*, DUMMY.DATA.SET2.*, 
CATALOG=DBT.USER.CATALOG
/*
```

A sample of the recovery report follows:

```
DSN.MULTIVOL.SAMPLE
  VOLUME: DSNS00 DSNS01
DB2DSN1.BSDS01.DATA
  VOLUME: DSNS00
DB2DSN1.BSDS01.INDEX
  VOLUME: DSNS00
DB2DSN1.BSDS02.DATA
  VOLUME: DSNS01
DB2DSN1.BSDS02.INDEX
  VOLUME: DSNS01
DB2DSN11.USER.CATALOG
  VOLUME: DSNS00
DB2DSN11.USER.CATALOG.CATINDEX
  VOLUME: DSNS00
```

RBR can provide a much more detailed report on dataset location and size as a part of its normal backup process.
**DB2 image copy considerations**

Traditional DB2 image copies may be necessary to guarantee recoverability of some DB2 objects. When backing up table spaces with 32 KB pages using an array-based backup and without doing a log suspend, it is possible that those objects could incur a broken page condition.

DB2 utilities that are run with LOG(NO) cannot be recovered using RECOVER LOGONLY methods and therefore should be backed up using either a traditional image copy or volume copy backup after performing LOG(NO) operations. Other image copy considerations involve the coordination of SYSLGRNX and SYSCOPY. The following describes the special circumstances that could impact DB2 recovery.

**Considerations for 32 KB pages**

For table spaces with 32 KB pages (also 16 KB pages if 3380 devices are used), under certain conditions, the array-based backup might contain broken pages that cannot be used as a base for a RECOVER LOGONLY recovery. This could happen when a table space has non-contiguous extents and the array copy is created at a point where an I/O to the first extent has completed, but the I/O to the rest of the page that resides on the second extent has not yet been issued. The probability that this will happen is small. It is possible to reduce the probability even further by forcing a DB2 checkpoint a few minutes prior to creating the volume copy. To ensure recoverability of these table spaces, take traditional DB2 image copies occasionally of table spaces with 32 KB pages.

A new DSNZPARM parameter DSVCI was introduced in DB2 V8. When set to YES, all new datasets for objects with page size > 4k are allocated with a CI size that is equal to the page size, thus eliminating the broken page exposure. However, after changing DSVCI to YES, existing objects with 32 KB pages have to be reorganized to recreate the underlying dataset with the matching CI. When this is accomplished, array-based backups can be created using EMC consistency technology without having a 32 KB page broken page exposure.
SYSLGRNX and SYSCOPY recovery

Care must be exercised when removing recovery information with the MODIFY RECOVERY utility. Recoveries from array-based backups may require archive logs created since DB2 was shut down normally, prior to the creating the backup. DSNDB01.SYSLGRNX entries from that same time are also required. Entries older than that can be deleted from DSNDB01.SYSLGRNX. Entries are deleted from DSNDB01.SYSLGRNX only if it appears that they are not required for recovery, based on information from DSNDB06.SYSCOPY.

When the MODIFY RECOVER utility cleans up recovery information, it deletes outdated image copy entries from DSNDB06.SYSCOPY and the related entries from DSNDB01.SYSLGRNX. If no entries are deleted from DSNDB06.SYSCOPY, no entries are deleted from DSNDB01.SYSLGRNX. If cleanup of DSNDB01.SYSLGRNX is desired, traditional image copies should be done occasionally to facilitate DSNDB01.SYSLGRNX cleanup.

Operations with LOG NO considerations

After performing a DB2 operation, such as LOAD or REORG TABLESPACE with LOG NO, perform a full DB2 image copy of the table space, or take an array-based backup of the environment. This ensures recoverability of the affected table spaces. The objects will be in copy pending state after the LOG NO operation anyway. So no updates past the LOG NO operation will be allowed until the copy is made.

Backing up DB2 objects with dataset snap

EMC technology can be used to back up and recover objects within a DB2 system. dataset snap can be used to back up DB2 objects. The dataset snap is much quicker than host data movers, and the size of the datasets does not affect the duration of the backup. However, very large ERP environments, where there are many thousands of table spaces are best served using volume replication rather than dataset snap.

Figure 30 shows how objects are backed up using dataset snap.
The following steps are required to perform a data snap backup of a DB2 table space:

1. Start table space for read-only access.
2. Issue a `QUIESCE WRITE(Y)` to ensure that all changed pages are flushed to disk by the buffer manager.
3. Snap all datasets associated with the table space to be backed up.
4. Start table space for read and write access.

It is possible to back up DB2 objects (tables, indexes) using TimeFinder volume processing. This procedure is different from backing up an entire DB2 system since only those volumes containing the objects to be backed up are copied. This technique is depicted in Figure 31 on page 153.
The following steps need to be executed when using this technique:

1. Do an **ESTABLISH** for TimeFinder/Mirror or **CREATE** with **pre-copy** for TimeFinder/Clone, depending on the TimeFinder product being used.

2. Start table spaces as **READ ONLY**.

3. Issue a DB2 **QUIESCE** WRITE(Y) to ensure that all changed pages are flushed to disk by the buffer manager.

4. Issue a TimeFinder/Mirror **SPLIT** or **ACTIVATE**.

5. Start table spaces RW to allow normal processing to resume.

This technique requires a very detailed understanding of the complete layout of the DB2 table spaces such that all components of the table spaces that are needed are backed up. If volumes or datasets are missed, any subsequent restore will fail. This technique might be applicable where only a portion of the DB2 system needs to be backed up and not the whole system. Full-volume restore with this technique is not recommended due to the conflicts expected with the ICF catalog.
Before selecting a backup method, some consideration should be given to the environment being backed up. There may be performance trade-offs if using virtual devices to create temporary backups. Table 5 is a comparison of the backup methods described in this chapter.

### Table 5  TimeFinder backup methods comparison

<table>
<thead>
<tr>
<th></th>
<th>Mirror</th>
<th>Full-volume clone</th>
<th>Virtual device</th>
<th>Dataset snap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real copy of data</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>required</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage cost</td>
<td>Full copy</td>
<td>Full Copy</td>
<td>Fractional</td>
<td>Full copy</td>
</tr>
<tr>
<td>Target will have long</td>
<td>Recommended</td>
<td>Recommended</td>
<td>Not recommended</td>
<td>Recommended</td>
</tr>
<tr>
<td>retention</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance Impact on</td>
<td>None</td>
<td>COA(^a), COFW(^b)</td>
<td>COFW(^b)</td>
<td>Minimal</td>
</tr>
<tr>
<td>Source data updated</td>
<td>Highly recommended</td>
<td>Highly recommended</td>
<td>Not recommended</td>
<td>Recommended</td>
</tr>
<tr>
<td>frequently</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Target availability</td>
<td>Delayed</td>
<td>Instantly</td>
<td>Instantly</td>
<td>Instantly</td>
</tr>
</tbody>
</table>

---

\(^a\) COA is Copy On Access. When tracks need to be read from the source volume because they are not on the target, they will be copied to the target.

\(^b\) COFW is Copy On First Write. The first time a protected track is written to it must be copied to the target volume or VDEV. This impact is negligible after Enginuity 5772.
Symmetrix arrays with DB2 system point-in-time recovery

DB2 V8 introduced new functionality called system point-in-time recovery, which is enabled using two new utilities: `BACKUP SYSTEM` and `RESTORE SYSTEM`. These new utilities invoke HSM and FlashCopy to execute backup and restore operations, using the underlying storage array. FlashCopy makes instantaneous volume-to-volume copies of the DB2 system in an operation that is very similar to TimeFinder/Clone full-volume snap. The following three sections describe the overall z/OS architecture necessary to make use of the `BACKUP SYSTEM` and `RESTORE SYSTEM` utilities, and the two different configurations for running FlashCopy operations on EMC Symmetrix arrays.

System point-in-time recovery overview

In order to use the system point-in-time recovery tools, DB2 V8 and V9 require that two SMS copy pools be defined: Log copy pool and database copy pool. The log copy pool is defined for the BSDS and logs. The database copy pool is defined for everything else. The names of these two copy pools are `DSN$LOCN$LG` for the logs and `DSN$LOCN$DB` for all the other data, where `LOCN` is the location of the DB2 system. This naming convention is required for the utilities to work.

In addition to this segregation, it is recommended to place the ICF catalogs that reference each set of data on the volumes containing the data. This makes life much easier during a restore process, as the ICF catalog is restored back to the point-in-time of the volume copy, ensuring it actually matches the datasets it references.

When a `BACKUP SYSTEM` utility runs, HSM invokes FlashCopy through DFDSS to create one of two types of backups. `FULL` (default) causes both the log and database copy pools to be backed up. `DATA ONLY` creates a backup of only the database copy pool.

*Figure 32* and *Figure 33* show examples of the two types of backups.
Figure 32  DB2 BACKUP SYSTEM FULL (default)

A BACKUP SYSTEM utility is executed causing HSM to invoke FlashCopy to replicate the entire DB2 system to the copy pool backup volumes.
A `BACKUP SYSTEM DATAONLY` utility is executed causing HSM to invoke FlashCopy to backup only the volumes in the DSN$LOCN$DB copy pool.

**Compatible Flash for z/OS and DB2**

Compatible Flash for z/OS is a host-based solution for providing FlashCopy support for Symmetrix subsystems in a z/OS environment. Compatible Flash intercepts and interprets IBM FlashCopy channel commands on the z/OS host and emulates command execution on EMC Symmetrix storage systems. Compatible Flash for z/OS does not run on Enginuity 5773 or later.

The technologies that enable Compatible Flash for z/OS are:

- EMC TimeFinder/Clone Mainframe SNAP Facility
- EMC ResourcePak Base for z/OS
Compatible Flash for z/OS runs as an environment within the EMC Symmetrix Control Facility (EMCSCF) and is enabled through a licensed feature code (LFC). When this product is licensed and enabled, the DB2 system point-in-time recovery utilities can be used if they are available.

Compatible Native Flash for Mainframe

Compatible Native Flash for Mainframe is a Symmetrix Enginuity product that was released with 5773 microcode. This licensed product emulates all FlashCopy channel commands within the Enginuity operating environment. No host software is necessary for this feature. DB2 **BACKUP SYSTEM** and **RESTORE SYSTEM** utilities can be used, if they are available, when this feature is licensed and enabled in the Symmetrix controller.

SYSPITR with TimeFinder

It is possible to create a backup with EMC TimeFinder technology that can be used by the DB2 **RESTORE SYSTEM** utility. Refer to the Section “Combining SYSPITR with TimeFinder” on page 177 for the procedure to do the restore from this backup. Any of the procedures outlined in the Section “Making a copy of the DB2 volumes” on page 138 would work as long the DB2 log is suspended when the copy is made.

Back up DB2 V9 for object-level recovery

In DB2 version 9, IBM introduced object-level recovery from a system-level backup. To make this functionality available, the system parameter `SYSTEM-LEVEL BACKUPS` must be set to `YES` on panel DSNTIP6, so that the **RECOVER** utility considers system-level backups. If the system-level backup resides on DASD, it is used for the restore of the object. If the system-level backup no longer resides on DASD, and has been dumped to tape, then the dumped copy is used for the restore of the object if FROMDUMP is specified on the **RECOVER** utility.
DB2 V10 introduced the capability to back up a table space using FlashCopy. This feature is called FlashCopy Image Copy (FCIC). The output from the copy utility using this feature is a VSAM dataset and not a QSAM dataset, as you might expect. This allows the utility to make a consistent copy of a set of table spaces together by rolling back uncommitted changes in the table spaces after they have been copied.

FlashCopy Image Copy is supported by all versions of EMC compatible FlashCopy software.
Space Efficient FlashCopy

FlashCopy SE, or Space Efficient FlashCopy is a feature very similar to TimeFinder Snap with virtual devices. The target volumes do not occupy any space until they are established with a source volume in a full-volume FlashCopy relationship. Changed tracks on the source volume are stored in an area called a repository on a DS8xxx storage controller. Changed tracks on the target are also stored in the repository. The repository is analogous to the SAVDEV area for TimeFinder/Snap.

Currently, EMC provides support FlashCopy SE operations using thin devices as the FlashCopy targets. FlashCopy NOCOPY operations using thin devices as target volumes is equivalent to a FlashCopy SE function. To implement space-efficient FlashCopy operations with DB2, place thin devices in the SMS copy pools and set VERSIONS=0 on the copy pool definition itself.

EMC’s implementation of space efficient FlashCopy has the following advantages:

- Flexible configurations—Any thin device in the array can be used as a FlashCopy target.
- Incremental copies—Only the changed tracks since the last FlashCopy operation are copied.
- Both classic FlashCopy and FlashCopy SE are allowed using a thin device.
- The thin pool/repository can be dynamically increased if necessary.
- The FlashCopy operation can be converted from NOCOPY to COPY.
- You can FlashCopy a dataset to a thin device.

Since the thin volumes are not known to z/OS as SE volumes, the IDCAMS report using the keywords “LISTDATA SPACEEFFICIENT” does not return any data for thin volumes. In a similar fashion, both TSO and ICKDSF queries for space efficient devices also result in no volumes being reported.
Rocket Backup and Recovery for DB2 on z/OS

The design and configuration of a backup solution using array-based functions requires some effort to manage and maintain on an on-going basis. Some of the functions that have to be performed in working closely with the storage administrators in the organization are:

- Determining UCBs for source volumes
- Determining UCBs for target volumes (BCVs, clones, or VDEVs)
- Establishing the pairings using JCL
- Validating completeness of backup
- Offloading the target volumes to sequential media
- Tracking the backups (locations and RBAs)

In the case of recovery, complex JCL must be generated for restore and perform forward recovery for many of the following functions:

- Determining RBA/LRSN point of recovery
- Determining location of appropriate backup (for the chosen RBA/LRSN)
- Removal of ICF Catalog allocations on the source volumes
- Creation of restore JCL for (potentially) many volumes
- Object restore from a DB2 system backup
- Generation of RECOVERY LOGONLY statements
- Generation of REBUILD INDEX statements (if necessary).

Recognizing that these operations were somewhat demanding on DBA time, EMC teamed with a partner company called Rocket Software to develop an integrated solution that provides all this functionality and more.

This solution is called Rocket Backup and Recovery for DB2 on z/OS and was written specifically for TimeFinder functionality. It supports TimeFinder/Mirror, TimeFinder/Clone and TimeFinder/Snap. It has since been extended to support FlashCopy, and has recently gained a lot of traction in the industry as it is now included in IBM’s DB2 Recovery Expert for z/OS product.
This chapter describes the ways that DB2 can be restored and recovered from Symmetrix array-based backups.

- Overview ........................................................................................................ 164
- TimeFinder restore and parallel recovery ................................................. 165
- DB2 system restart/recovery ...................................................................... 165
- Optimizing DB2 recovery ........................................................................... 169
- Recovering DB2 objects ............................................................................. 172
- DB2 System Point-in-Time Recovery tools ............................................. 176
- Rocket Backup and Recovery for z/OS .................................................... 178
- DB2 steps for recovery in detail ............................................................... 179
Overview

When a copy of a DB2 system, DB2 volumes, or DB2 datasets has been made using TimeFinder, a number of options are available to process that copy depending on the nature of the restore and/or recovery action required. What is possible is determined by how the backup was made. Table 6 lists at a high level what is possible in terms of recovery for a given backup scenario.

<table>
<thead>
<tr>
<th>How the backup was created</th>
<th>What kind of restore/recovery is available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full system backup with log suspended</td>
<td>• Full system restore to copy</td>
</tr>
<tr>
<td></td>
<td>• Full system restore and forward recovery</td>
</tr>
<tr>
<td></td>
<td>• Dataset recovery</td>
</tr>
<tr>
<td>Full system backup without log suspended (using ECA)</td>
<td>• Full system restore to copy</td>
</tr>
<tr>
<td></td>
<td>• Full system restore and forward recovery (RBR only)</td>
</tr>
<tr>
<td></td>
<td>• Dataset recovery</td>
</tr>
<tr>
<td>Full system backup with \texttt{BACKUP SYSTEM} utility</td>
<td>• \texttt{RESTORE SYSTEM} and forward recovery</td>
</tr>
<tr>
<td></td>
<td>• \texttt{RESTORE SYSTEM LOGONLY} (when using TimeFinder)</td>
</tr>
<tr>
<td></td>
<td>• \texttt{RESTORE SYSTEM LOGONLY} (when using FRRECOV)</td>
</tr>
<tr>
<td></td>
<td>• Dataset recovery</td>
</tr>
<tr>
<td>Partial system backup (volume level)</td>
<td>• Volume restore (ICF catalogs must be managed) and mandatory forward recovery for all datasets</td>
</tr>
<tr>
<td></td>
<td>• Dataset restore and mandatory forward recovery</td>
</tr>
<tr>
<td>Dataset backup (tablespace quiesced)</td>
<td>• Dataset restore</td>
</tr>
<tr>
<td></td>
<td>• Dataset recovery</td>
</tr>
<tr>
<td>Dataset backup (tablespace not quiesced)</td>
<td>• Dataset and mandatory forward recovery</td>
</tr>
</tbody>
</table>

In addition, it should be noted that the required copy of the data to be restored may no longer be on disk, as it may have been offloaded to tape or other sequential media.
Note: In most of the array-based backup solutions, entries are not placed in SYSIBM.SYSCOPY, and thus some of the specific functions of the RECOVER command are not available.

TimeFinder restore and parallel recovery

When a TimeFinder restore is initiated, the data is immediately available on the source volumes even though all of the changed tracks have not yet been written back to the source devices from the target devices. If a track that has not been restored is accessed on the source volume, it is copied on demand from the target volume. DB2 recovery processes can begin as soon as the TimeFinder RESTORE command has been issued. This parallelism significantly reduces recovery time.

DB2 system restart/recovery

In order to perform a full system restart and recovery, it is necessary to have performed a full system backup at some time in the past. Additionally, it is assumed that a full system backup has been performed using one of the following techniques:

- TimeFinder/Mirror
- TimeFinder/Clone
- TimeFinder/Snap
- FlashCopy
- BACKUP SYSTEM command (system point-in-time recovery tool)

Rocket Backup and Recovery for z/OS can also provide for both full system backup and restore, but since the procedures with this product are fully automated, they are not covered in this section. Rather a quick overview of the RBR functionality is provided in the section “Rocket Backup and Recovery for z/OS” on page 178.

At this point, a decision needs to be made if the system is going to be restored to the point of the copy, or whether it is going to be restored and rolled forward from the point in time of the copy to a specific RBA or LRSN. The procedures for the two requirements are very much different and are described separately in the following two sections.
Regardless of the two techniques chosen, it is mandatory that the ICF catalog for the datasets on the volumes being restored is also returned to the point of time of the backup. If this doesn’t happen there will likely be differences between the volume information for the dataset in the ICF catalog and where the datasets actually are, on the restored volumes. If the ICF catalog itself is on the restored volumes, as recommended in the section “Database organization to facilitate recovery” on page 132, this concern is avoided.

**DB2 system restore to copy**

In this process the system is restored in its entirety, which means logs, BSDS, DB2 catalog, DB2 directory and all table spaces. When using this technique it is not possible to roll forward the database to a point in time subsequent to the time the backup was taken. Figure 34 on page 166 depicts the complete restore of all volumes in a full system restore. The restore process is similar regardless of which array-based copy methodology is being used.
In order to accomplish this process, there are a number of steps to be taken. These are:

1. Shut down the DB2 system on the source—See the section “Shutting down the DB2 system” on page 179.

2. Vary off the source volumes — see the section “Vary off the DB2 source volumes” on page 179.

3. Perform the system restore function using Symmetrix array capability (TimeFinder or FlashCopy)—See the section “TimeFinder system restore function” on page 180. If the required volume backup is on tape, use the procedure described in the section “System restore from tape” on page 182.

4. Vary on the source volumes—See the section “Vary on the DB2 source volumes” on page 183.

5. Bring up the DB2 system.

**DB2 system restore and roll-forward**

In this process the system data structures are restored (not including the active logs, archive logs and BSDS). Then the database is rolled forward to a selected RBA or LRSN, using the `RECOVER LOGONLY` utility. Figure 35 on page 168 depicts the restore of all volumes that comprise the data structures in the system. The restore process is similar regardless of which array-based copy methodology is being used.

**NOTICE**

In order for this procedure the recovery structures and data structures must be separated, as described in “Database organization to facilitate recovery” on page 132. If this recommendation is not followed, the following procedures will not work.
Figure 35  **Restore of system data structures (no log or BSDS)**

In order to accomplish the restore and roll forward of the logs, there are a number of steps to be taken. These are:

1. Shut down the DB2 system on the source—See the section “Shutting down the DB2 system” on page 179.

2. Determine the RBA to which the system should be recovered. To do so, analyze the output from DB2 utilities DSNJU004 and DSN1LOGP. This is used as the ENDRBA or the ENDLRSN of the conditional restart in step 5. If data sharing, all members must have the same ENDLRSN.

3. Find a backup that was taken prior to this RBA.

4. Vary off the source volumes—See the section “Vary off the DB2 source volumes” on page 179.

5. Perform the volume restore function for only the data structures—See the section “TimeFinder system restore function” on page 180. If the required volume backup is on tape, use the procedure described in the section “System restore from tape” on page 182.
6. Vary on the source volumes—See the section “Vary on the DB2 source volumes” on page 183.

7. Add a conditional restart record to the BSDS, as described in the section “Create a conditional restart record” on page 184.

8. Perform a `RESTORE SYSTEM LOGONLY` as shown in the section “Using SYSPITR to restore the DB2 system” on page 185. If the DB2 system point-in-time recovery tools are not available, the database objects must be recovered individually. First, perform a `RECOVER LOGONLY` on the DB2 catalog and directory (in the correct order as documented in the Recovering catalog and directory objects section of the DB2 Utility Guide and Reference). Second, recover the rest of the environment (table spaces and indexes) with LOGONLY, and rebuild indexes if necessary (those defined with COPY NO).

9. Shut down DB2 and restart with the normal DSNZPARM.

After completing the previous steps, the state of the database is exactly as if DB2 had crashed and been restarted at the time of the specified RBA/LRSN.

---

**Optimizing DB2 recovery**

When the system point-in-time recovery tools are not available, all objects must be recovered individually. Using this method, the `RECOVER LOGONLY` process causes DB2 to read the HPGRBRBA recovery base RBA from the header page of each page set. Then, in one pass of the log, every update to a page set that is detected is reapplied to the appropriate page set. Although not all the page sets may have had updates since the array copy was created, DB2 must perform a dynamic allocation for every dataset, read the header page, and determine the starting point for the log apply. When the frequency of creating the array copy is high, most of the page sets probably have no updates at all.

Performing `RECOVER LOGONLY` for each page set may take a long time if a large DB2 system is being recovered, even if there were few updates only to a few page sets since the array-based backup environment was created. To make the recovery process more efficient, use the following procedure to limit the set of objects to be recovered to updated page sets only. This can reduce the recovery
time from hours to minutes. Actual recovery time depends on the
volatility of the data in the system and the size of the log range to be
processed.

To optimize DB2 system recovery, follow these steps:

1. Determine the RBA to which the system should be recovered. To
do so, analyze the output from DB2 utilities DSNJU004 and
DSN1LOGP. This is used as the ENDRBA or ENDLRSN of the
conditional restart in step 8.

2. Find a backup that was taken prior to this RBA or LRSN.

3. Run a DSNJU004 report and find the starting RBA or LRSN of the
third checkpoint before the time of the backup found in step 2.

4. Run a DSN1LOGP with an RBASTART=(RBA from step 3),
RBAEND=(RBA from step 1), and SUMMARY (ONLY). Use the
following sample JCL:

```
//STEP1 EXEC PGM=DSN1LOGP
//SYSSUMRY DD SYSOUT=*  
//SYSPRINT DD SYSOUT=*  
//BSDS DD DSN=DB2DSN1.BDS01,DISP=SHR  
//SYSIN DD * 
  RBASTART (RBA_from_step_3) RBAEND (RBA_from_step_1)  
  SUMMARY (ONLY)  
 OR 
  LRSNSTART (LRSN_from_step_3) LRSNEND 
  (LRSN_from_step_1)  
  SUMMARY (ONLY)  
/*
```

This report lists the completed and running events in this log
range, including modified page sets. These modified page sets are
the page sets that need to be recovered.

5. Vary off the source volumes — see the section “Vary off the DB2
source volumes” on page 179.

6. Perform a restore of the data structures as described in the section
“TimeFinder system restore function” on page 180.

7. Vary on the source volumes — see the section “Vary on the DB2
source volumes” on page 183.

8. Add a conditional restart record to the BSDS as described in the
section “Create a conditional restart record” on page 184.

9. Bring up the DB2 system with DEFER ALL — see the section
“Bring up DB2 in DEFER ALL mode” on page 184.
10. Perform `RECOVER LOGONLY` (in the correct order as documented in the `Recovering Catalog and Directory Objects` section of the `DB2 Utility Guide and Reference` manual) only for the DB2 catalog and directory objects that are in the list created in step 4.

11. Recover the rest of the page sets from the list created in step 4 (table spaces and indexes) with `LOGONLY`; rebuild indexes if necessary (defined with `COPY NO`).

12. Shut down DB2 and restart with the normal `DSNZPARM`. 
Recovering DB2 objects

DB2 objects can be recovered from either a backup residing on a BCV, on a standard device, on a virtual device, or other backup devices, such as tape. The procedures to follow when recovering a DB2 object vary depending on the mechanism used to create the backup. The following sections describe the various procedures that can be utilized and the actions that must be taken.

Object restore from system backup

After an array-based copy of a DB2 system has been taken, it is possible to restore individual objects from that copy. This section discusses the various techniques that can be used to retrieve datasets from a volume-based backup and provides sample JCL for those activities.

It should be noted that all objects restored in this manner must be recovered using RECOVER LOGONLY processing to a point in time that guarantees that the object is transactionally consistent. It is the user's responsibility to return the object (or set of objects) to this consistency point. In DB2 9, the RECOVER LOGONLY performs an additional activity to guarantee that the object being recovered is returned to a COMMIT boundary. This guarantees that no partial transaction updates are applied to any given object. As long as all related objects in the transaction are restored to the same RBA/LRSN, transactional integrity can be achieved using this technique.

Restoring objects using TimeFinder/Snap

TimeFinder/Snap can be used to restore a DB2 object if the dataset still resides on target volume. In reality, it does not matter which TimeFinder product was used to create the target volume. The target volume containing the dataset backup can be offline or online when using the following procedure and JCL.

1. Stop the table spaces requiring recovery.
2. Perform a dataset snap from the backup device to the source device. All datasets that are related to the object(s) to be recovered should be copied with dataset snap. The following JCL is an example of TimeFinder Snap dataset with a volume list that includes all the UCBs of the target volumes that hold the backup.
For this JCL setup, the “backup” UCBs do not need to be online, as TimeFinder/Snap can process the volumes while they are offline.

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//SYSOUT DD SYSOUT=*
//QCOUTPUT DD SYSOUT=*
//QCINPUT DD *
DEFINE SOURCE_VOLUME_LIST DBR1VOLS -
   (UNIT(4000-4007) )
SNAP DATASET ( -
   SOURCE(DBRD.DSNDBC.DBNAME.TSNAME.I0001.A001) -
   TARGET(DBRD.DSNDBC.DBNAME.TSNAME.I0001.A001) -
   SOURCE_VOLUME_LIST(DBR1VOLS) -
   REUSE(YES) -
   REPLACE(YES) -
   TOLERATE_REUSE_FAILURE(Y) -
   )
/*
Note: With this JCL setup, it is not necessary to know which volumes actually contain the datasets to be snapped. TimeFinder/Snap performs the necessary work to discover the location of the datasets.
*/
```

3. Start the stopped table space for utility access:
   `-START DB(DBNAME) SP(TSNAME) ACCESS(UT)

4. Perform a RECOVER LOGONLY, as described in the section “Dataset recover logonly” on page 186.

5. Rebuild any indexes that were defined with COPY(NO).

6. Start the stopped table spaces for read/write access:
   `-START DB(DBNAME) SP(TSNAME) ACCESS(RW)

---

**Full-volume restore for database objects**

TimeFinder/Mirror, full-volume snap or virtual device snap can be used to restore the database or groups of objects of one application running under a DB2 system. This requires precise isolation of the objects on their respective volumes. The following are the requirements for using this method of backing up and restoring DB2 objects:

1. Volumes must be dedicated to the application data. All application datasets must be on this set of volumes.
2. All dedicated ICF user catalogs must be on the same set of volumes.

3. No datasets are created, deleted, or renamed subsequent to the TimeFinder full-volume copy.

4. **RECOVER LOGONLY** must be performed unless objects are quiesced at the time of the copy (when a **REPAIR SET LEVELID** can be used).

If all the above requirements are met, then the application could be restored using a TimeFinder/Mirror restore operation, TimeFinder/Clone full-volume snap back or a virtual device restore; but all the volumes in the group must be restored.

---

**Restoring objects from tape**

In some cases, recovery from disk is not possible because the volume copy containing the restore point has been dumped to tape, and the TimeFinder targets no longer hold the desired copy of the DB2 object that need to be recovered. In this case, the restore/recovery from tape can be accomplished using DFDSS or FDR.

**Restoring objects using DFDSS**

To perform a logical restore of a single dataset from a full-volume dump on tape using DFDSS, follow these steps:

1. Stop the table space.

2. Use IDCAMS to delete the VSAM cluster.

3. Perform a dataset restore using the following JCL:

   ```plaintext
   //STEP1 EXEC PGM=ADDRSSU,REGION=6M
   //SYSPRINT DD SYSOUT=* 
   //SYSIN DD *
   COPY DATASET ( 
     INCLUDE ( 
       DBT1D.DSNDBC.DBNAME.TSNAME.I0001.A001 - 
       DBT1D.DSNDBC.DBNAME.ISNAME.I0001.A001 - 
     ) 
     OUTDYNAM( DBR005 ) 
     PHYSINDY( BKP005 ) 
     FR( PREF ) TOl(ENQF) REPLACE 
     ALLEXCP ALLDATA(*) 
   )
   */
   ```
If a full-volume dump to tape has already been performed, and the disk copy of the data is not available or is out of date, a selected restore of the dataset from a full-volume dump on tape can be executed using the following JCL (as an example):

```jcl
//STEP1 EXEC PGM=ADRDSSU,REGION=6M
//TF011 DD <TAPE_ALLOCATION>
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
RESTORE DATASET ( 
   INCLUDE ( 
      DBRD.DSNDBC.DBNAME.TSNAME.I0001.A001 
      ,DBRD.DSNDBC.DBNAME.ISNAME.I0001.A001 
   ) 
   OUTDYNAM( DB2001 ) 
   INDDNAME( TF011 ) 
   TOL(ENQF) REPLACE
/*
Note: If the dataset that needs to be restored is a multi-volume dataset then all the components on each volume must be restored individually and to different volumes from the ones that originally held the volumes.
*/
```

4. Perform an IDCAMS DEFINE RECATALOG for the restored cluster, as shown in the section “Define recatalog of a DB2 dataset” on page 185.

5. Start the stopped table space for utility access:

   -START DB(DBNAME) SP(TSNAME) ACCESS(UT)

6. Perform a RECOVER LOGONLY, as described in the section “Dataset recover logonly” on page 186.

7. Rebuild any indexes that were defined with COPY(NO) or any that were not restored and recovered.

**Dataset restore using FDR**

To perform a logical restore of a single dataset from a full-volume dump on tape using FDR, follow these steps:

1. Stop the table space.

2. Perform a dataset restore using the following JCL:

   ```jcl
   //FDRREST EXEC PGM=FDRDSF
   //SYSPRINT DD SYSOUT=* 
   //TAPE1 DD DSN=DB2DSN1.FULLBKUP.DSNT00, 
   // UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(1,SL), 
   // VOL=SER=TAPE00
   ```
//DISK1 DD UNIT=3390,DISP=SHR,VOL=SER=DSNS00

Note: If the dataset that needs to be restored is a multi-volume dataset then all the components on each volume must be restored individually and to different volumes from the ones that originally held the volumes.

3. Start the stopped table space for utility access:
   
   `START DB(DBNAME) SP(TSNAME) ACCESS(UT)`

4. Perform a `RECOVER LOGONLY` as described in the section “Dataset recover logonly” on page 186.

5. Rebuild any indexes that were defined with `COPY(NO)` or any that were not restored and recovered.

6. Start the stopped table space(s) for read/write access:

   `-START DB(DBNAME) SP(TSNAME) ACCESS(RW)`

---

**DB2 System Point-in-Time Recovery tools**

The DB2 System Point-in-Time Recovery tools were first introduced in DB2 V8 and later enhanced in DB2 V9 and V10. These tools allow the user to back up the whole DB2 system by making use of the storage array replication capability. Initially, the sweet spot for the tool was large ERP systems with tens of thousands of DB2 objects, which made image copies a non-viable backup solution. Being able to back up thousands of datasets in a matter of seconds not only helps DBAs by a reduction of complexity but also provides for an easily achievable point of consistency for the application across all table spaces.

The `BACKUP SYSTEM` functionality in the DB2 point-in-time recovery tools requires that the underlying storage array supports FlashCopy commands. In an EMC Symmetrix subsystem, FlashCopy support is provided in two separate and mutually exclusive ways. These are described in the following two sections.
EMC Compatible Flash for z/OS

Compatible Flash for z/OS is for customers using a DMX-3 or earlier who wish to perform FlashCopy operations on the Symmetrix array. FlashCopy channels commands from the host are intercepted by the SCF address space and converted to TimeFinder/Clone full-volume snap commands. The SCF address space then returns control to the host program with control structures exactly as if the FlashCopy channel commands were executed. Users wishing to use the system point-in-time recovery tools for DB2 on a DMX-3 or earlier, using Enginuity 5773 or earlier, can use Compatible Flash for z/OS. There are no changes necessary from the DB2 aspect to enable this. From the z/OS aspect the software needs to be licensed and installed in the host.

EMC Compatible Native Flash for Mainframe

Compatible Native Flash for Mainframe (or Compatible Native Flash) is a compatible implementation of the IBM FlashCopy feature. When Compatible Native Flash is licensed and enabled on a Symmetrix (DMX-4 system and later), FlashCopy channel commands received from the mainframe host are acted upon by the Symmetrix subsystem and control is returned back to the host at channel program completion. The emulation of FlashCopy on the array is transparent to DB2. Thus all the system point-in-time recovery mechanisms can be executed on a Symmetrix system with Compatible Native Flash licensed and enabled.

Combining SYSPITR with TimeFinder

While the DB2 system point-in-time recovery tools are specifically designed to use FlashCopy to make volume copies, there are circumstances where they can be used with TimeFinder copies of volumes. For instance the RESTORE SYSTEM LOGONLY functionality can be used after a TimeFinder restore of the data structures, as described in the section “TimeFinder system restore function” on page 180.

One must be careful in this TimeFinder process and steps taken to perform this action since the BACKUP SYSTEM command executes more steps than just simply copying the source volumes with FlashCopy. One of the additional functions that takes place during a
**DB2 Recovery Procedures**

**BACKUP SYSTEM** is that the recovery base log point (**HPGRBLP** field in the **DBD01** header page) is updated for the start of the log replay. This is not updated when a TimeFinder backup is executed.

This process works with TimeFinder when either of the following is true:
- The log is suspended when making the TimeFinder copy.
- RBR with ECA was used to created the TimeFinder copy.

Both of these processes update the **HPGRBLP** field in **DBD01** to create a starting point for the **RESTORE SYSTEM LOGONLY**.

---

**Rocket Backup and Recovery for z/OS**

Rocket Backup and Recovery for z/OS (RBR) was written specifically for EMC Symmetrix arrays and includes all the functionality to perform restores from array-based backups. In addition, if the backup is not available on disk, it keeps the metadata about the sequential media containing the backup and dynamically allocates it, if needed. RBR provides the ability to use an ECA consistent array-based copy made by TimeFinder/Mirror, TimeFinder/Clone, or TimeFinder/Snap and perform system restore and roll forward of the logs. RBR can also restore objects from a system backup if that requirement is defined in the backup profile.

RBR provides the following features for recovery:
- Integration with TimeFinder/Mirror, TimeFinder/Clone, and TimeFinder/Snap through API calls
- Automatic creation of all restore and recovery JCL
- Automatic creation of conditional restart JCL
- Automatic starting and stopping of databases, table spaces, and index spaces
- Automatic VARYING ON and OFF UCBs as and when required
- Automatic removal of Catalog Address Space (CAS) allocations on target volumes
- Automatic generation of rebuild index control parameters for COPY=NO indexes
- Location of correct backup image (whether on disk or on tape) through use of an RBA or LRSN (data sharing)
DB2 steps for recovery in detail

The following sections are the detailed explanations of the steps that are explained in high level in the earlier sections of this chapter. This should be used for reference only when guided by the numbered processes explained previously.

Shutting down the DB2 system

When restoring and recovering a whole DB2 system from a volume-based copy it is always necessary to shut down the source DB2 system. This is accomplished by the z/OS command:

```
:ssid stop db2
```

where ssid is the DB2 system identifier.

Check to see that all the DB2 address spaces are stopped before continuing the procedure. In rare cases, you may have to cancel these address spaces to bring down DB2.

Vary off the DB2 source volumes

Before performing the array-based volume restore, you must vary off the DB2 source volumes. Using the configuration depicted in Figure 27 on page 133, you would issue the z/OS command:

```
V 1000-1007,OFFLINE
```

When using a sysplex environment for data sharing, make sure the volumes are offline to all LPARs. This can be accomplished by using the **RO *ALL** syntax and with the **VARY** command.

Check to see if the volumes are in fact offline by issuing the following command:

```
D U,,,1000,8
```
If the volumes are shown to be OFFLINE-P, then they may have a ICF catalog allocation against them. To see the catalog allocations currently retained by the Catalog Address Space (CAS), issue the following command:

F CATALOG,OPEN

If there are any allocations on the source volumes, they can be removed by executing:

F CATALOG,UNCATALOG(<icf_catalog_name>)

---

**TimeFinder system restore function**

The process of performing a TimeFinder system restore from a disk copy is very similar whether using TimeFinder/Mirror, TimeFinder/Clone, or TimeFinder/Snap. The main difference is the JCL and the command syntax. Note that if forward recovery is needed, the restore process should only restore the data structures and this requires adherence to the organization described in Figure 27 on page 133.

**Restoring with TimeFinder/Mirror**

When using TimeFinder/Mirror, the following JCL should be executed to restore both the recovery structures and the data structures:

```jcl
//TFRESTOR EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD *
GLOBAL MAXRC=4,WAIT
RESTORE 01,4000-4007
SPLIT 02,4000-4007
/*
```

If forward recovery is required, only the data structures must be restored:

```jcl
//TFRESTOR EXEC PGM=EMCTF
//SYSOUT DD SYSOUT=* 
//SYSIN DD *
GLOBAL MAXRC=4,WAIT
RESTORE 01,4004-4007
SPLIT 02,4004-4007
/*
```
Restoring with TimeFinder/Clone

When using TimeFinder/Clone, there is no restore function as such. The “restore” is enabled by copying the target volumes back over the source volumes in a reverse of the initial operation. The reverse restore functionality is available with TimeFinder/Clone only on distributed platforms. The following JCL should be executed to copy back the target volumes from onto the source volumes:

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//STEPLIB DD DSN=SYSx.SSNP580.LINKLIB,DISP=SHR
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSDSN=*
//QCOUTPUT DD SYSDSN=*  
//QCINPUT DD *
SNAP VOLUME -
   (  
      SOURCE (UNIT(4000-4007)) - 
      TARGET (UNIT(1000-1007)) -  
   ) 
ACTIVATE  
/*

Note that if group processing was used to create the target volumes this (non-grouped) procedure is still used and recommended since the DB2 system is made available earlier in the process, and there is no need, in this case, to use pre-copy.

If forward recovery is required, only the data structures must be restored:

```
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSDSN=*
//QCOUTPUT DD SYSDSN=*  
//QCINPUT DD *
SNAP VOLUME -
   (  
      SOURCE (UNIT(4004-4007)) - 
      TARGET (UNIT(1004-1007)) -  
   ) 
ACTIVATE  
/*

Restoring with TimeFinder/Snap

When using TimeFinder/Snap to restore the DB2 system, it is important to ensure that the PERSISTENT(YES) keyword is used. This allows all other existing SNAPs to be kept during the restore. If the initial SNAP restore does not fix the problem that is trying to be avoided, another SNAP instance can then be selected.
The following JCL should be executed to copy back the VDEV volumes onto the source volumes:

```plaintext
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=* 
//QCOUTPUT DD SYSOUT=* 
//QCINPUT DD *
RESTORE VOLUME ( 
   VDEV (UNIT(4000-4007)) 
      TO (UNIT(1000-1007)) 
      AUTOMATIC_CLEANUP(YES) 
      PERSISTENT(YES) 
   ) 
ACTIVATE 
/*

After the ACTIVATE has completed, the restore is, in effect, finished, since the copying process on the array takes place in the background.

If forward recovery is required, only the data structures must be restored:

```plaintext
//EMCSNAP EXEC PGM=EMCSNAP,REGION=0M
//EMCGROUP DD DSN=HLQ.GROUP.PDS,DISP=SHR
//SYSOUT DD SYSOUT=* 
//QCOUTPUT DD SYSOUT=* 
//QCINPUT DD *
RESTORE VOLUME 
   ( 
      VDEV (UNIT(4004-4007)) 
      TO (UNIT(1004-1007)) 
   ) 
ACTIVATE 
/*
```

---

**System restore from tape**

A recovery from tape may be required to achieve a point of consistency that no longer exists on a disk copy. In this case, DFDSS may be used to restore the source volumes from tape to a prior point in time. Though the source units need to be online when restoring from tape, it is still a good practice to vary them off, and then on, to make sure they are not being used by any process.

To perform a full restore of a volume from tape to a source DB2 volume, use the following JCL:

```plaintext
//RESTORE EXEC PGM=ADRDSSU 
//SYSPRINT DD SYSOUT=* 
```
//SYSOUT DD SYSOUT=*  
//DISK1 DD UNIT=3390,DISP=SHR,VOL=SER=DSNS00  
//TAPE1 DD DSN=DB2BKP.FULLBKUP.DSNT00,  
//                UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(1,SL),  
//                VOL=SER=TAPE00  
//DISK2 DD UNIT=3390,DISP=SHR,VOL=SER=DSNS01  
//TAPE2 DD DSN=DB2BKP.FULLBKUP.DSNT01,  
//                UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(1,SL),  
//                VOL=SER=TAPE01  
//SYSIN DD *  
//          RESTORE INDD(TAPE1) OUTDD(DISK1)  
//          RESTORE INDD(TAPE2) OUTDD(DISK2)  
/*

If the entire environment is restored (including the ICF user catalogs), no further steps are required. However, if the ICF catalog is not restored, then an IDCAMS DEFINE RECATALOG for the cluster page sets on the restored volumes is required to synchronize the ICF user catalogs and the actual volume contents. This is a very time-consuming and complex process. An example of why this may be required is covered in the section “Recovering DB2 objects” on page 172.

To perform a full restore of a volume from tape to a DB2 source volume using FDR, use the following JCL:

//RESTORE EXEC PGM=FDRDSF  
//SYSPRINT DD SYSOUT=*  
//DISK1 DD UNIT=3390,VOL=SER=DSNS00,DISP=OLD  
//TAPE1 DD DSN=DB2BKP.FULLBKUP.DSNT00,  
//                UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(1,SL),  
//                VOL=SER=TAPE00  
//DISK2 DD UNIT=3390,VOL=SER=DSNS01,DISP=OLD  
//TAPE2 DD DSN=DB2BKP.FULLBKUP.DSNT01,  
//                UNIT=TAPE,DISP=(OLD,KEEP),LABEL=(1,SL),  
//                VOL=SER=TAPE01  
//SYSIN DD *  
//          RESTORE TYPE=FDR  
/*

Vary on the DB2 source volumes

When using array-based restore functionality, the restored volumes can be used after they have been varied online to the LPAR. Based on the configuration depicted in Figure 27 on page 133, the following z/OS command can be used to vary on the volumes to the LPAR:

V 1000-1007,ONLINE
Check to see if the volumes are, in fact, online by issuing the following command:

```
D U,,1000,7
```

---

### Create a conditional restart record

The conditional restart record is used to control the next restart of DB2. The RBA or LRSN (data-sharing) is specified for the target point in time for the `RECOVER LOGONLY` process. In the following two JCL examples, `xxxxxxxxxxxxx` represents the RBA/LRSN point to which recovery is needed.

There are two forms of conditional restart records used in these processes. The first form is for `RESTORE SYSTEM LOGONLY` execution and involves the use of the system point-in-time recovery tool. The second form is for all other cases.

**System point-in-time recovery tool JCL:**

```
//DSNTLOG EXEC PGM=DSNJU003
//SYSUT1 DD DISP=OLD,DSN=DBSL.DBS1.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DBSL.DBS1.BSDS02
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
  
  CRESTART
  CREATE,SYSPITR=xxxxxxxxxxxxx,FORWARD=YES,BACKOUT=YES
  *
```

**All other recoveries:**

```
//DSNTLOG EXEC PGM=DSNJU003
//SYSUT1 DD DISP=OLD,DSN=DBSL.DBS1.BSDS01
//SYSUT2 DD DISP=OLD,DSN=DBSL.DBS1.BSDS02
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *
  
  CRESTART
  CREATE,ENDRBA=xxxxxxxxxxxxx,FORWARD=YES,BACKOUT=YES
  or
  CRESTART
  CREATE,ENDLRSN=xxxxxxxxxxxxx,FORWARD=YES,BACKOUT=YES
  *
```

---

### Bring up DB2 in DEFER ALL mode

When restarting DB2 for forward recovery, DB2 must be brought up in DEFER ALL mode. This can be accomplished by having a separate ZPARM module that has been compiled with DEFER ALL in the
DSN6SPRM macro and using it when DB2 is brought up. DEFER ALL defers restart log apply processing for all objects, including DB2 catalog and directory objects.

DEFER does not affect processing of the log during restart. Therefore, even if DEFER ALL is specified, DB2 still processes the full range of the log for both the forward and backward log recovery phases of restart. However, logged operations are not applied to the dataset.

DB2 resolves all units of work that were outstanding at the time of the recovery by creating compensation records in the log. When the system is recovered with the LOGONLY option, all log records are applied, including the compensation records. A WTOR is issued, warning that this is a cold start. Respond y to the WTOR.

Using SYSPITR to restore the DB2 system

The following JCL can be used to perform a full system recovery after all the data structures have been restored using a volume-based copy technique. This feature makes use of the DB2 system point-in-time recovery tool:

```
//STEP01 EXEC DSNUPROC,UTPROC='',SYSTEM='DBS1',TIME=1440
//SYSIN DD *
   RESTORE SYSTEM LOGONLY
/*

It should be noted that if the data structures have been copied using TimeFinder for the backup purpose, then for this process to work with the TimeFinder array copy, the recovery base log point (HPGRBLP field in the DBD01 header page) must be updated at the time of the backup. See the section “Combining SYSPITR with TimeFinder” on page 177 for more details.

Define recatalog of a DB2 dataset

Perform an IDCAMS DEFINE RECATALOG to the restored cluster dataset. The following is a JCL example for performing this task:

```
//RECAT EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//SYSIN DD *
   DEFINE CLUSTER -
     ( NAME(DBSD.DSNDBC.DBCNAME.TSNAME.I0001.A001) -
       VOL(DBS001) -
       RECATALOG LINEAR) -
     DATA -
```
Dataset recover logonly

To perform a RECOVER LOGONLY on an object, you can use the following JCL. This rolls forward the object to a specific RBA xxxxxxxxxx:

```bash
//RCVRLOGO EXEC   PGM=DSNUTILB,
  REGION=006M,COND=(4,LT),PARM=(DBS1)
//STEPLIB DD      DISP=SHR,DSN=DB2.DBS1.SDSNLOAD
// DD            DISP=SHR,DSN=DB2.DBS1.SDSNEXIT
//SYSPRINT DD     SYSOUT=*  
//SYSPUT DD       SYSOUT=*  
//UTPRINT DD      SYSOUT=*  
//SYSIN DD        RECOVER
  TABLESPACE DBNAME.TSNAME
    TOLOGPOINT X'xxxxxxxxxxxxx'
    LOGONLY
    LOCALSITE

/*
This chapter presents the following topics:

- Overview ................................................................. 188
- Disaster recovery vs. disaster restart ......................... 189
- Definitions ................................................................. 190
- Design factors for disaster recovery and disaster restart 192
- Tape-based solutions .................................................. 198
- Remote replication challenges .................................... 200
- Array-based remote replication .................................. 204
- Planning for array-based replication ......................... 205
- SRDF/S single Symmetrix array to single Symmetrix array 205
- SRDF/S and Consistency Groups ............................... 207
- SRDF/A ................................................................. 214
- SRDF/AR single-hop .................................................. 217
- SRDF/AR Multi-hop .................................................... 219
- SRDF/Star .............................................................. 222
- SRDF/Extended Distance Protection ......................... 223
- DB2 data sharing restart considerations ...................... 225
- High-availability solutions ........................................ 227
Overview

A critical part of managing a database is planning for unexpected loss of data. The loss can occur from a disaster like fire or flood, or it can come from hardware or software failures. It can even come through human error or malicious intent. In each event, the DB2 system must be restored to some usable point, before application services can be resumed.

The effectiveness of any plan for restart or recovery involves answering the following questions:

◆ How much downtime is acceptable to the business?
◆ How much data loss is acceptable to the business?
◆ How complex is the solution?
◆ Are multiple recovery sites required?
◆ Does the solution accommodate the data architecture?
◆ How much does the solution cost?
◆ What disasters does the solution protect against?
◆ Is there protection against logical corruption?
◆ Is there protection against physical corruption?
◆ Is the database restartable or recoverable?
◆ How easily can the solution be tested?
◆ If failover happens, will failback work?

All restart and recovery plans include a replication component. In its simplest form, the replication process may be as easy as making a tape copy of the database and application. In a more sophisticated form, it could be real-time replication of all changed data to some remote location. Remote replication of data has its own challenges centered around:

◆ Distance
◆ Propagation delay (latency)
◆ Network infrastructure
◆ Data loss
This section provides an introduction to the spectrum of disaster recovery and disaster restart solutions for DB2 systems on EMC Symmetrix arrays.

**Disaster recovery vs. disaster restart**

Disaster recovery usually involves the use of backup technology in which data is copied to tape, and then it is shipped off site. When a disaster is declared, the remote site copies are restored and logs are applied to bring the data to a point of consistency. Once all recoveries are completed, the data is validated to ensure that it is correct. Coordinating to a common business point of recovery across all applications and other platforms can be difficult, if not impossible, using traditional recovery methods.

Disaster restart solutions allow the restart of all participating data resource managers to a common point of consistency utilizing the automated application of recovery logs during application initialization. The restart time is comparable to the length of time required for the application to restart after a power failure. A disaster restart solution can include other data, such as MQ Series persistent queues, VSAM and DB2 datasets, flat files, and so on, and is made possible with EMC technology. Restartable images that are dependent-write consistent can be created locally, and then transported to off site storage site by way of tape or SRDF to be used for disaster restart. These restartable images can also be created remotely. Dependent-write consistency is ensured by EMC technology, and transactional consistency is ensured by the DBMS at restart, similar to recovery from a local power failure.

Transactional consistency is made possible by the dependent I/O philosophies inherent in logging DBMS systems. Dependent-write I/O is the methodology all logging DBMS systems use to maintain integrity. Data writes are dependent on a successful log write in these systems, and therefore, restartability is guaranteed. Restarting a database environment, instead of recovering it, enhances availability and reduces the recovery time objective (RTO). The time required to do a traditional DB2 recovery is significantly greater than that required to restart a DB2 system at the recovery site.

The entire DB2 system can be restored at the volume level, and then restarted instead of recovered. EMC Consistency technology ensures that the DB2 system is consistent, from a dependent-write
perspective. DB2, upon restart, handles backing out of incomplete transactions. This method of restart is significantly faster than the traditional recovery methods.

EMC supports synchronous and asynchronous restart solutions. Asynchronous remote replication solutions are distinguished from synchronous remote replication solutions at the application or database level. With synchronous replication, the application waits for an acknowledgment from the storage unit that the remote data write has completed. Application logic does not allow any dependent-write activity until this acknowledgment has been received. If the time to receive this notice is elongated, then response times for users or applications can increase.

The alternative is to use an asynchronous remote replication solution that provides acknowledgement to the application that the write has been completed, regardless of whether the remote physical write activity has completed. The application can then allow dependent-write activity to proceed immediately.

It is possible to lose consistency and/or data in an asynchronous remote replication solution, so various techniques have been developed in the industry to prevent data loss and/or ensure data consistency. EMC uses SRDF Consistency Groups, TimeFinder Consistency Groups, SRDF/AR, SRDF/A, and SRDF/MSC individually or in combination to address this issue. Refer to Chapter 2, “EMC Foundation Products,” for further information on EMC products that support this kind of replication.

Definitions

In the following sections, the terms dependent-write consistency, database restart, database recovery, and forward recovery are used. A clear definition of these terms is required to understand the context of this chapter.

**Dependent-write consistency**

A dependent-write I/O is one that cannot be issued until a related predecessor I/O has completed. Dependent-write consistency is a data state where data integrity is guaranteed by dependent-write I/Os embedded in application logic. Database management systems practice the use of dependent-write consistency.
Database management systems must devise protection against abnormal termination in order to successfully recover from one. The most common technique used is to guarantee that a dependent write cannot be issued until a predecessor write has completed. Typically, the dependent write is a data or index write, while the predecessor write is a write to the log. Because the write to the log must be completed prior to issuing the dependent data write, the application thread is synchronous to the log write, that is, it waits for the log write to complete before continuing. The result of this kind of strategy is a dependent-write consistent database.

**Database restart**

Database restart is the implicit application of database logs during its normal initialization process to ensure a transactionally consistent data state.

If a database is shut down normally, the process of getting to a point of consistency during restart requires minimal work. If the database abnormally terminates, then the restart process takes longer, depending on the number and size of in-flight transactions at the time of termination. An image of the database created by using EMC consistency technology while it is running, without conditioning the database, is in a dependent-write consistent data state, which is similar to that created by a local power failure. This is also known as a DBMS restartable image. The restart of this image transforms it to a transactionally consistent data state by completing committed transactions and rolling back uncommitted transactions during the normal database initialization process.

**Database recovery**

Database recovery is the process of rebuilding a database from a backup image, and then optionally, explicitly applying subsequent logs to roll the data state forward to a designated point of consistency. Database recovery is only possible with databases configured with archive logging.

Creating a recoverable DB2 system copy on z/OS can be performed in one of three ways:

1. Using traditional backup utilities.
2. With the database quiesced and copying the database components using external tools.

3. Using storage replication capabilities:
   a. With the database in SET LOG SUSPEND mode and copying the database using external tools.
   b. Quiesced using external tools.
   c. Running using consistency technology to create a restartable and recoverable image.

---

**Forward recovery**

With DB2, it is possible to take a DBMS restartable image of the database and apply subsequent archive logs to roll forward the database to a point in time after the image was created. This means that the image created can be used in a backup strategy in combination with archive logs.

---

**Design factors for disaster recovery and disaster restart**

Loss of data or loss of application availability has a varying impact from one business type to another. For instance, the loss of transactions for a bank could cost millions whereas system downtime may not have a major fiscal impact. On the other hand, businesses that are primarily web-based may require 100 percent application availability in order to survive. The two factors, loss of data and loss of uptime are the business drivers that define the baseline requirements for a DR solution. When quantified, these two factors are more formally known as recovery point objective (RPO) and recovery time objective (RTO), respectively.

When evaluating a solution, the RPO and RTO requirements of the business need to be met. In addition, the solution needs to take into consideration operational complexity, cost, and the ability to return the whole business to a point of consistency. Each of these aspects is discussed in the following sections.
Recovery point objective

The RPO is a point of consistency to which a user wants to recover or restart. It is measured in the amount of time from when the point of consistency was created or captured to the time the disaster occurred. This time equates to the acceptable amount of data loss. Zero data loss (no loss of committed transactions from the time of the disaster) is the ideal goal, but the high cost of implementing such a solution must be weighed against the business impact and cost of a controlled data loss.

Some organizations, like banks, have zero data loss requirements. The database transactions entered at one location must be replicated immediately to another location. This can have an impact on application performance when the two locations are far apart. On the other hand, keeping the two locations close to one another might not protect against a regional disaster, like a power outage or a hurricane.

Defining the required RPO is usually a compromise between the needs of the business, the cost of the solution and the risk of a particular event happening.

Recovery time objective

The RTO is the maximum amount of time allowed for recovery or restart to a specified point of consistency. This time involves many factors. The time taken to:

- Provision power, utilities, and so on
- Provision hosts with the application and database software
- Configure the network
- Restore the data at the new site
- Roll forward the data to a known point of consistency
- Validate the data

Some delays can be reduced or eliminated by choosing certain DR options, like having a hot site where hosts are preconfigured and on standby. Also, if storage-based replication is used, the time taken to restore the data to a usable state is almost completely eliminated.
As with RPO, each solution for RTO has a different cost profile. Defining the RTO is usually a compromise between the cost of the solution and the cost to the business when database and applications are unavailable.

**Operational complexity**

The operational complexity of a DR solution may be the most critical factor in determining the success or failure of a DR activity. The complexity of a DR solution can be considered as three separate phases:

1. Initial setup of the implementation.
2. Maintenance and management of the running solution, including testing.
3. Execution of the DR plan in the event of a disaster.

While initial configuration and operational complexities can be a demand on people resources, the third phase, execution of the plan, is where automation and simplicity must be the focus. When a disaster is declared, key personnel may not be available, in addition to the loss of hosts, storage, networks, buildings, and so on. If the complexity of the DR solution is such that skilled personnel with an intimate knowledge of all systems involved are required to restore, recover, and validate application and database services, the solution has a high probability of failure.

Multiple database environments grow organically over time into complex federated database architectures. In these federated database environments, reducing the complexity of DR is absolutely critical. Validation of transactional consistency within the complex database architecture is time consuming, costly, and requires intimate knowledge of the application. One of the reasons for this complexity is due to the heterogeneous databases and operating systems in these federated environments. Across multiple heterogeneous platforms, it is hard to establish a common clock and therefore hard to determine a business point of consistency across all platforms. This business point of consistency has to be created from intimate knowledge of the transactions and data flows.
Source host activity

DR solutions may or may not require additional processing activity on the source z/OS hosts. The extent of that activity can impact both response time and throughput of the production application. This effect should be understood and quantified for any given solution to ensure the impact to the business is minimized. The effect for some solutions is continuous while the production application is running. For other solutions, the impact is sporadic, where bursts of write activity are followed by periods of inactivity.

Production impact

Some DR solutions delay the host activity while taking actions to propagate the changed data to another location. This action only affects write activity, and although the introduced delay may only be of the order of a few milliseconds, it can impact response time in a high-write environment. Synchronous solutions introduce delay into write transactions at the source site, asynchronous solutions do not.

Target host activity

Some DR solutions require a target z/OS host at the remote location to perform DR operations. This host has both software and hardware costs and needs personnel with physical access to it for basic operational functions, like power on and power off. Ideally, this host could have some usage like running development or test databases and applications. Some DR solutions require more target host activity than others and some require none.

Number of copies of data

DR solutions require replication of data in one form or another. Replication of a DB2 system and associated datasets can be as simple as making a tape backup and shipping the tapes to a DR site or as sophisticated as asynchronous array-based replication. Some solutions require multiple copies of the data to support DR functions. More copies of the data may be required to perform testing of the DR solution in addition to those that support the DR process.
Distance for solution

Disasters, when they occur, have differing ranges of impact. For instance, a fire may take out a building, an earthquake may destroy a city, or a hurricane may devastate a region. The level of protection for a DR solution should address the probable disasters for a given location. For example, when protecting against an earthquake, the DR site should not be in the same locale as the production site. For regional protection, the two sites need to be in two different regions. The distance associated with the DR solution affects the kind of DR solution that can be implemented.

Bandwidth requirements

One of the largest costs for DR is in provisioning bandwidth for the solution. Bandwidth costs are an operational expense; this makes solutions which have reduced bandwidth requirements very attractive to customers. It is important to estimate in advance the bandwidth consumption of a given solution to be able to anticipate the running costs. Incorrect provisioning of bandwidth for DR solutions can have an adverse affect on production performance and can invalidate the overall solution.

Federated consistency

Databases are rarely isolated islands of information with no interaction or integration with other applications or databases. Most commonly, databases are loosely and/or tightly coupled to other databases using triggers, database links, and stored procedures. Some databases provide information downstream for other databases using information distribution middleware. Other databases receive feeds and inbound data from message queues and EDI transactions. The result can be a complex interwoven architecture with multiple interrelationships. This is referred to as a federated database architecture.

With a federated database architecture, making a DR copy of a single database without regard to other components invites consistency issues and creates logical data integrity problems. All components in a federated architecture need to be recovered or restarted to the same dependent-write consistent point of time to avoid these problems.
With this in mind, it is possible that point database solutions for DR, like log shipping, do not provide the required business point of consistency in a federated database architecture. Federated consistency solutions guarantee that all components, databases, applications, middleware, flat files, and so on, are recovered or restarted to the same dependent-write consistent point in time.

Testing the solution

Tested, proven, and documented procedures are also required for a DR solution. Many times the DR test procedures are operationally different from a true disaster set of procedures. Operational procedures need to be clearly documented. In the best case scenario, companies should periodically execute the actual set of procedures for DR. This could be costly to the business because of the application downtime required to perform such a test, but it is necessary to ensure validity of the DR solution.

Ideally, DR test procedures should be executed by personnel not involved with creating the procedures. This helps ensure the thoroughness and efficacy of the procedures.

Cost

The cost of doing DR can be justified by comparing it to the cost of not doing it. What does it cost the business when the database and application systems are unavailable to users? For some companies this is easily measurable, and revenue loss can be calculated per hour of downtime or per hour of data loss.

Whatever the business, the DR cost is going to be an extra expense item and, in many cases, with little in return. The costs include, but are not limited to:

- Hardware (storage, hosts, and maintenance)
- Software licenses and maintenance
- Facility leasing/purchase
- Utilities
- Network infrastructure
- Personnel
Tape-based solutions

When it is not feasible to implement an SRDF solution, tape-based solutions are alternatives that have been used for many years. The following sections describe tape-based disaster recovery and restart solutions.

Tape-based disaster recovery

Traditionally, the most common form of disaster recovery was to make a copy of the database onto tape, and using PTAM (Pickup Truck Access Method), take the tapes off site to a hardened facility. In most cases, the database and application needed to be available to users during the backup process. Taking a backup of a running database created a “fuzzy” image of the database on tape, one that required database recovery after the image had been restored. Recovery usually involved application of logs that were active during the time the backup was in process. These logs had to be archived and kept with the backup image to ensure successful recovery.

The rapid growth of data over the last two decades has meant that this method has become unmanageable. Making a “hot” copy of the database is now the standard, but this method has its own challenges. How can a consistent copy of a DB2 system and supporting datasets be made when they are changing throughout the duration of the backup? What exactly is the content of the tape backup at completion? The reality is that the tape data is a fuzzy image of the disk data, and considerable expertise is required to restore the database back to a database point of consistency.

In addition, the challenge of returning the data to a business point of consistency, where a particular database must be recovered to the same point as other databases or applications, is making this solution less viable.

Tape-based disaster restart

Tape-based disaster restart is a recent development in disaster recovery strategies and is used to avoid the fuzziness of a backup taken while the database and application are running. A “restart” copy of the system data is created by locally replicating the disks that contain the production data, and creating a dependent-write consistent point-in-time image of the source disks. This image is a
DBMS restartable image as described earlier. Thus, if this image were restored and the database brought up, the database would perform an implicit recovery to attain transactional consistency.

The restartable image on the disks can be backed up to tape and moved off site to a secondary facility. The frequency that the backups are created and shipped off site determines the amount of data loss with this solution. Other circumstances, such as bad tapes, could also dictate how much data loss is incurred at the remote site.

The time taken to restore the database is a factor to consider since reading from tape is typically slow. Consequently, this solution can be effective for customers with relaxed RTOs.
Remote replication challenges

Replicating database information over long distances for the purpose of disaster recovery is challenging. Synchronous replication over distances greater than 200 km may not be feasible due to the negative impact on the performance of writes because of propagation delay. However, some form of asynchronous replication must be adopted for long distance replication. Considerations in this section apply to all forms of remote replication technology, whether they are array-based, host-based or managed by the database.

Remote replication solutions usually start with initially copying a full DB2 system image to the remote location. This is called instantiation of the system. There are a variety of ways to perform this. After instantiation, only the changes from the source site are replicated to the target site in an effort to keep the target up to date. Some methodologies may not send all of the changes (certain log shipping techniques for instance), by omission rather than design. These methodologies may require periodic re-instantiation of the database at the remote site.

The following considerations apply to remote replication of databases:

- Propagation delay (latency due to distance)
- Bandwidth requirements
- Network infrastructure
- Method and frequency of instantiation
- Change rate at the source site
- Locality of reference
- Expected data loss
- Failback considerations

Propagation delay

Electronic operations execute at the speed of light. The speed of light in a vacuum is 186,000 miles per second. The speed of light through glass (in the case of fiber optic media) is less, approximately 115,000 miles per second. In other words, in an optical network like SONET for instance, it takes 1 millisecond to send a data packet 125 miles or 8
milliseconds for 1000 miles. All remote replication solutions need to be designed with a clear understanding of the propagation delay impact.

**Bandwidth requirements**

All remote replication solutions have some bandwidth requirements because the changes from the source site must be propagated to the target site. The more changes there are, the greater the bandwidth that is needed. It is the change rate and replication methodology that determine the bandwidth requirement, not necessarily the size of the database.

Data compression can help reduce the quantity of data transmitted and therefore the size of the pipe required. Certain network devices, like switches and routers, provide native compression, some by software and some by hardware. GigE directors provide native compression in an SRDF pairing between Symmetrix systems. The amount of compression achieved is dependent on the type of data that is being compressed. Typical character and numeric database data compresses at about a 2 to 1 ratio. A good way to estimate how the data compresses is to assess how much tape space is required to store the database during a full backup process. Tape drives perform hardware compression on the data prior to writing it. For instance, if a 300 GB database takes 200 GB of space on tape, the compression ratio is 1.5 to 1.

For most customers, a major consideration in the disaster recovery design is cost. It is important to recognize that some components of the end solution represent a capital expenditure and some an operational expenditure. Bandwidth costs are operational expenses, and thus any reduction in this area, even at the cost of some capital expense, is highly desirable.

**Network infrastructure**

The choice of channel-extension equipment, network protocols, switches, routers and so on, ultimately determines the operational characteristics of the solution. EMC has a proprietary BC Design Tool to assist customers in analysis of the source systems and to determine the required network infrastructure to support a remote replication solution.
Understanding Disaster Restart and Disaster Recovery

Method and frequency of instantiation

In all remote replication solutions, a common requirement is for an initial, consistent copy of the complete DB2 system to be replicated to the remote site. The initial copy from source to target is called instantiation of the database at the remote site. Following instantiation, only the changes made at the source site are replicated. For large databases, sending only the changes after the initial copy is the only practical and cost-effective solution for remote database replication.

In some solutions, instantiation of the database at the remote site uses a process that is similar to the one that replicates the changes. Some solutions do not even provide for instantiation at the remote site (log shipping for instance). In all cases, it is critical to understand the pros and cons of the complete solution.

Method of re-instantiation

Some techniques to perform remote replication of a database require periodic refreshing of the remote system with a full copy of the database. This is called re-instantiation. Technologies such as log-shipping frequently require this since not all activity on the production database may be represented in the log. In these cases, the disaster recovery plan must account for re-instantiation and also for the fact that there may be a disaster during the refresh. The business objectives of RPO and RTO must likewise be met under those circumstances.

Change rate at the source site

After instantiation of the database at the remote site, only changes to the database are replicated remotely. There are many methods of replication to the remote site and each has differing operational characteristics. The changes can be replicated using logging technology, hardware and software mirroring for example. Before designing a solution with remote replication, it is important to quantify the average change rate. It is also important to quantify the change rate during periods of burst write activity. These periods might correspond to end of month/quarter/year processing, billing, or payroll cycles. The solution needs to be designed to allow for peak write workloads.
Locality of reference

Locality of reference is a factor that needs to be measured to understand if there will be a reduction of bandwidth consumption when any form of asynchronous transmission is used. Locality of reference is a measurement of how much write activity on the source is skewed. For instance, a high locality of reference application may make many updates to a few tables in the database, whereas a low locality of reference application rarely updates the same rows in the same tables during a given period of time.

It is important to understand that while the activity on the tables may have a low locality of reference, the write activity into an index might be clustered when inserted rows have the same or similar index column values, rendering a high locality of reference on the index components.

In some asynchronous replication solutions, updates are “batched” up into periods of time and sent to the remote site to be applied. In a given batch, only the last image of a given row/block is replicated to the remote site. So, for highly skewed application writes, this results in bandwidth savings. Generally, the greater the time period of batched updates, the greater the savings on bandwidth.

Log shipping technologies do not take into account locality of reference. For example, a row updated a 100 times, is transmitted 100 times to the remote site, whether the solution is synchronous or asynchronous.

Expected data loss

Synchronous DR solutions are zero data loss solutions, that is to say, there is no loss of committed transactions from the time of the disaster. Synchronous solutions may also be impacted by a rolling disaster in which case work completed at the source site after the rolling disaster started may be lost. Rolling disasters are discussed in detail in a later section.

Non-synchronous DR solutions have the potential for data loss. How much data is lost depends on many factors, most of which have been defined in the previous sections. The quantity of data loss that is expected for a given solution is called the recovery point objective (RPO). For asynchronous replication, where updates are batched and sent to the remote site, the maximum amount of data lost is two
cycles or two batches worth. The two cycles that may be lost include the cycle currently being captured on the source site and the one currently being transmitted to the remote site. With inadequate network bandwidth, data loss could increase due the increase in transmission time.

Failback considerations

If there is the slightest chance that failover to the DR site may be required, then there is a 100 percent chance that failback to the primary site is also required, unless the primary site is lost permanently. The DR architecture should be designed in such a way as to make failback simple, efficient and low risk. If failback is not planned for, there may be no reasonable or acceptable way to move the processing from the DR site, where the applications may be running on tier-2 systems and tier-2 networks and so on, back to the production site.

In a perfect world, the DR process should be tested once a quarter, with database and application services fully failed over to the DR site. The integrity of the application and database needs to be verified at the remote site to ensure that all required data was copied successfully. Ideally, production services are brought up at the DR site as the ultimate test. This means that production data would be maintained on the DR site, requiring a failback when the DR test completed. While this is not always possible, it is the ultimate test of a DR solution. It not only validates the DR process, but also trains the staff on managing the DR process should a catastrophic failure ever occur. The downside for this approach is that duplicate sets of hosts and storage need to be present in order to make an effective and meaningful test. This tends to be an expensive proposition.

Array-based remote replication

Customers can use the capabilities of a Symmetrix storage array to replicate the database from the production location to a secondary location. No host CPU cycles are used for this, leaving the host dedicated to running the production application and database. In addition, no host I/O is required to facilitate this, the array takes care of all replication, and no hosts are required at the target location to manage the target array.

EMC provides multiple solutions for remote replication of databases:
SRDF/S: Synchronous SRDF
SRDF/A: Asynchronous SRDF
SRDF/AR: SRDF Automated Replication
SRDF/Star: Software that enables concurrent SRDF/S and SRDF/A operations from the same source volumes.

Each of these solutions is discussed in detail in the following sections. In order to use any of the array-based solutions, it is necessary to coordinate the disk layout of the databases with this kind of replication in mind.

Planning for array-based replication

All Symmetrix solutions replicating data from one location to another are volume based. This allows the Symmetrix controller to be agnostic to applications, database systems, operating systems, and so on.

In addition, if a DB2 system is to be replicated independently of other DB2 systems, it should have its own set of dedicated devices. That is, the devices used by a DB2 system should not be shared with other applications or DB2 systems.

SRDF/S single Symmetrix array to single Symmetrix array

Synchronous SRDF, or SRDF/S, is a method of replicating production data changes from locations that are no greater than 200 km apart. Synchronous replication takes writes that are inbound to the source Symmetrix array and copies them to the target Symmetrix array. The write operation is not acknowledged as complete to the host until both Symmetrix arrays have the data in cache. It is important to realize that while the following examples involve Symmetrix arrays, the fundamentals of synchronous replication described here are true for all synchronous replication solutions. Figure 36 on page 206 shows this process.
The following steps describe Figure 36:

1. A write is received into the source Symmetrix cache. At this time, the host does not receive acknowledgement that the write is complete.

2. The source Symmetrix array uses SRDF/S to transmit the write to the target Symmetrix array.

3. The target Symmetrix array sends an acknowledgement back to the source that the write was received.

4. Channel end/device end is presented to the host.

These four steps introduce a delay in the processing of writes as perceived by the DBMS on the source host. This delay may or may not impact performance depending on the nature of the writes, synchronous or asynchronous. The amount of delay depends on the exact configuration of the network, the storage, the write block size, and the distance between the two locations. Note that reads from the source Symmetrix array are not affected by the replication.

Dependent-write consistency is inherent in a synchronous relationship as the target R2 volumes are at all times equal to the source provided that a single RA group is used. If multiple RA groups are used, or if multiple Symmetrix arrays are used to hold the source DB2 system, SRDF Consistency Groups (SRDF/CG) must be used to guarantee consistency. SRDF/CG is described in the following section.

Once the R2s in the group are visible to the host, the host can issue the necessary commands to access the devices, and they can then be varied online to the host. When the data is available to the host, the DB2 system can be restarted. DB2 performs an implicit recovery when started up for the first time. Transactions that were committed but not completed are rolled forward and completed using the
information in the active logs. Transactions that have updates applied to the database but were not committed are rolled back. The result is a transactionally consistent database.

SRDF/S and Consistency Groups

Zero data loss disaster recovery techniques tend to use straightforward database and application restart procedures. These procedures work well if all processing and data mirroring stop at the same instant in time at the production site, when a disaster happens. Such is the case when there is a site power failure.

However, in most cases, it is unlikely that all data processing ceases at an instant in time. Computing operations can be measured in nanoseconds, and even if a disaster takes only a millisecond to complete, many such computing operations could be completed between the start of a disaster until all data processing ceases. This gives us the notion of a rolling disaster. A rolling disaster is a series of events taken over a period of time that comprise a true disaster. The specific period of time that makes up a rolling disaster could be milliseconds (in the case of an explosion) or minutes in the case of a fire. In both cases the DR site must be protected against data inconsistency.

Rolling disaster

Protection against a rolling disaster is required when the data for a database resides on more than one Symmetrix array or multiple RA groups. Figure 37 on page 208 depicts a dependent-write I/O sequence where a predecessor log write is happening prior to a page flush from a database buffer pool. The log device and data device are on different Symmetrix arrays with different replication paths. Figure 37 on page 208 demonstrates how rolling disasters can affect this dependent-write sequence.
Understanding Disaster Restart and Disaster Recovery

Figure 37 Rolling disaster with multiple production Symmetrix arrays

1. This example of a rolling disaster starts with a loss of the synchronous links between the bottom source Symmetrix array and the target Symmetrix array. This prevents the remote replication of data on the bottom source Symmetrix array.

2. The bottom Symmetrix array, which is now no longer replicating, receives a predecessor log write of a dependent write I/O sequence. The local I/O is completed, but it is not replicated to the remote Symmetrix array, and the tracks are marked as being “owed” to the target Symmetrix array. Nothing prevents the predecessor log write from completing to the host completing the acknowledgement process.

3. Now that the predecessor log write has completed, the dependent data write is issued. This write is received on both the source Symmetrix array and the target Symmetrix array because the rolling disaster has not yet affected those communication links.

4. If the rolling disaster ended in a complete disaster, the condition of the data at the remote site is such that it creates a data-ahead-of-log condition, which is an inconsistent state for a database. The severity of the situation is that when the database is restarted, performing an implicit recovery, it may not detect the inconsistencies. A person extremely familiar with the transactions...
running at the time of the rolling disaster might be able to detect the inconsistencies. Database utilities could also be run to detect some of the inconsistencies.

A rolling disaster can happen in such a manner that data links providing remote mirroring support are disabled in a staggered fashion, while application and database processing continues at the production site. The sustained replication during the time when some Symmetrix arrays are communicating with their remote partners through their respective links while other Symmetrix arrays are not (due to link failures) can cause data integrity exposure at the recovery site. Some data integrity problems caused by the rolling disaster cannot be resolved through normal database restart processing and may require a full database recovery using appropriate backups and logs. A full database recovery elongates overall application restart time at the recovery site.

### Protecting against a rolling disaster

SRDF Consistency Groups (SRDF/CG) technology provides protection against rolling disasters. A consistency group is a set of Symmetrix volumes spanning multiple RA groups and/or multiple Symmetrix frames that replicate as a logical group to other Symmetrix arrays using Synchronous SRDF. It is not a requirement to span multiple RA groups and/or Symmetrix frames when using consistency groups. Consistency group technology guarantees that if a single source volume is unable to replicate to its partner for any reason, then all the volumes in the group stop replicating. This ensures that the image of the data on the target Symmetrix array is consistent from a dependent-write perspective.

Figure 38 on page 210 depicts a dependent-write I/O sequence where a predecessor log write is happening prior to a page flush from a database buffer pool. The log device and data device are on different Symmetrix arrays with different replication paths. Figure 38 on page 210 demonstrates how rolling disasters can be prevented using EMC consistency group technology.
Understanding Disaster Restart and Disaster Recovery

Figure 38  Rolling disaster with SRDF Consistency Group protection

1. Consistency group protection is defined containing volumes X, Y, and Z on the source Symmetrix array. This consistency group definition must contain all of the devices that need to maintain dependent-write consistency. A mix of CKD (mainframe) and FBA (UNIX/Windows) devices can be logically grouped together. In some cases, the entire processing environment may be defined in a consistency group to ensure dependent-write consistency.

2. The predecessor log write occurs to volume Z, causing a ConGroup trip.

3. A ConGroup trip holds the I/O that could not be replicated along with all of the I/O to the logically grouped devices. It is held long enough to issue two (2) I/Os per Symmetrix array. The first I/O puts the devices in a suspend-pending state. The second I/O performs the suspend of the R1/R2 relationship for the logically grouped devices, which immediately disables all replication to the remote site. This allows other devices outside of the group to continue replicating, provided the communication links are available.
4. After the R1/R2 relationship is suspended, all deferred write I/Os are released, allowing the predecessor log write to complete to the host. The dependent data write is issued by the DBMS and arrives at X but is not replicated to the R2(X).

5. If a complete failure occurred from this rolling disaster, dependent-write consistency at the remote site is preserved. If a complete disaster did not occur, and the failed links were activated again, the consistency group replication could be resumed once synchronous mode is achieved. It is recommended to create a copy of the dependent-write consistent image while the resume takes place. Once the SRDF process reaches synchronization the dependent-write consistent copy is achieved at the remote site.

SRDF/S with multiple source Symmetrix arrays

The implications of spreading a database across multiple Symmetrix frames or across multiple RA groups and replicating in synchronous mode were discussed in previous sections. The challenge in this type of scenario is to protect against a rolling disaster. SRDF Consistency Groups can be used to avoid data corruption in a rolling disaster situation.

Consider the architecture depicted in Figure 39 on page 212.
To protect against a rolling disaster, a consistency group can be created that encompasses all the volumes on all Symmetrix arrays participating in replication as shown by the blue dotted oval.

**ConGroup considerations for DB2 on z/OS**

Refer to DB2 Considerations for ConGroup in Chapter 2 for a list of the datasets required in a ConGroup for a synchronous restart solution. Figure 40 on page 213 shows an example of a DB2 ConGroup configuration.
Figure 40  DB2 ConGroup configuration
SRDF/A

SRDF/A, or asynchronous SRDF, is a method of replicating production data changes from one Symmetrix array to another using delta set technology. Delta sets are the collection of changed blocks grouped together by a time interval that can be configured at the source site. The default time interval is 30 seconds. The delta sets are then transmitted from the source site to the target site in the order they were created. SRDF/A preserves the dependent-write consistency of the database at all times at the remote site.

The distance between the source and target Symmetrix array is unlimited, and there is no host impact. Writes are acknowledged immediately when they reach the cache of the source Symmetrix array. SRDF/A is only available on the DMX and VMAX Family of Symmetrix controllers. Figure 41 depicts the process:

1. Writes are received into the source Symmetrix cache. The host receives immediate acknowledgement that the write is complete. Writes are then gathered into the capture delta set for 30 seconds.

2. A delta set switch occurs and the current capture delta set becomes the transmit delta set by changing a pointer in cache. A new empty capture delta set is created.

3. SRDF/A sends the changed blocks that are in the transmit delta set to the remote Symmetrix array. The changes collect in the receive delta set at the target site. When the replication of the
transmit delta set is complete, another delta set switch occurs and a new empty capture delta set is created with the current capture delta set becoming the new transmit delta set. The receive delta set becomes the apply delta set.

4. The apply delta set marks all the changes in the delta set against the appropriate volumes as invalid tracks and begins destaging the blocks to disk.

5. The cycle repeats continuously.

With sufficient bandwidth for the source database write activity, SRDF/A transmits all changed data within the default 30 seconds. This means that the maximum time the target data can be behind the source is 60 seconds (two replication cycles). At times of high write activity, it may not be possible to transmit all the changes that occur during a 30 second interval. This means that the target Symmetrix array falls behind the source Symmetrix array by more than 60 seconds. Careful design of the SRDF/A infrastructure and a thorough understanding of write activity at the source site are necessary to design a solution that meets the RPO requirements of the business at all times.

Consistency is maintained throughout the replication process on a delta set boundary. The remote Symmetrix array does not apply a partial delta set, which would invalidate consistency. Dependent-write consistency is preserved by placing a dependent write in either the same delta set as the write it depends on or a subsequent delta set.

Different command sets are used to enable SRDF/A depending on whether the SRDF/A group of devices is contained within a single Symmetrix array or is spread across multiple Symmetrix arrays.

---

**SRDF/A using a single source Symmetrix array**

Before the asynchronous mode of SRDF can be established, initial instantiation of the database has to have taken place. In other words, a baseline full copy of all the volumes that are going to participate in the asynchronous replication must be executed first. This is usually accomplished using the Adaptive Copy mode of SRDF.
SRDF/A using multiple source Symmetrix arrays

When a DB2 system is spread across multiple Symmetrix arrays and SRDF/A is used for long distance replication, separate software must be used to manage the coordination of the delta set boundaries between the participating Symmetrix arrays and to stop replication if any of the volumes in the group cannot replicate for any reason. The software must ensure that all delta set boundaries on every participating Symmetrix array in the configuration are coordinated to give a dependent-write consistent point-in-time image of the DB2 system.

SRDF/A multi-session consistency (MSC) provides consistency across multiple RA groups and/or multiple Symmetrix arrays. MSC is available on 5671 microcode and above with ResourcePak Base 5.5 and above. SRDF/A with MSC is supported by an SRDF host process that performs cycle-switching and cache recovery operations across all SRDF/A sessions in the group. This ensures that a dependent-write consistent R2 copy of the database can be created at the remote site at all times. The MSC environment is set up and controlled though SCF (ResourcePak Base), within which MSC commands are managed. SRDF must be running on all hosts that can write to the set of SRDF/A volumes being protected. At the time of an interruption (SRDF link failure, for instance), MSC analyzes the status of all SRDF/A sessions and either commits the last cycle of data to the R2 target or discards it.

For more details see the SRDF/A TechBook on EMC Online Support.

How to restart in the event of a disaster

In the event of a disaster when the source Symmetrix array is lost, database and application services must be run from the DR site. A host at the DR site is required for restart. Before restart can be attempted, the R2 devices must be write enabled. SRDF host component commands can be used to do this.

Once the volumes are varied on to the target host and write enabled, the DB2 system can be restarted. Transactions that were committed but not completed are rolled forward and completed using the information in the active logs. Transactions that have updates applied to the database, but not committed, are rolled back. The result is a transactionally consistent database.
SRDF/AR single-hop

SRDF Automated Replication, or SRDF/AR, is a continuous movement of dependent-write consistent data to a remote site using SRDF adaptive copy mode and TimeFinder technology. TimeFinder/Mirror or TimeFinder/Clone is used to create a dependent-write consistent point-in-time image of the data to be replicated in the local Symmetrix array. The local TimeFinder target volumes also have an R1 personality, which means that SRDF in Adaptive Copy mode can be used to replicate the data from these volumes to the target site. Since the R1s are not changing, replication completes in a finite length of time. The length of time for replication depends on the size of the network pipe between the two locations, the distance between the two locations, the quantity of changed data tracks, and the locality of reference of the changed tracks. On the remote Symmetrix array, another TimeFinder copy of the data is made using data on the R2s. This is necessary because the next SRDF/AR cycle replaces the R2 image in a non-ordered fashion, and if a disaster were to occur while the R2s were synchronizing, there would not be a valid copy of the data at the DR site. The remote copy of the R2s in the remote Symmetrix array is commonly called the gold copy of the data. The whole process then repeats.

With SRDF/AR, there is no host impact. Writes are acknowledged immediately when they reach the cache of the source Symmetrix array. Figure 42 shows this process using TimeFinder/Mirror for the local copies.

Figure 42  SRDF/AR single-hop replication configuration

The following steps describe Figure 42:
1. Writes are received into the source Symmetrix cache and are acknowledged immediately. The BCVs are already synchronized with the STDs at this point. A consistent SPLIT is executed against the STD-BCV pairing to create a PIT image of the data on the BCVs.

2. SRDF transmits the data on the BCV/R1s to the R2s in the remote Symmetrix array.

3. When the BCV/R1 volumes are synchronized with the R2 volumes, they are reestablished with the standards in the source Symmetrix array. This causes the SRDF links to be suspended. At the same time, an incremental establish is performed on the target Symmetrix array to create a gold copy on the BCVs in that frame.

4. The BCVs in the remote Symmetrix array are fully synchronized with the R2s, they are split, and the configuration is ready to begin another cycle.

5. The cycle repeats based on configuration parameters. The parameters can specify the cycles to begin at specific times, specific intervals or to run back to back.

It should be noted that cycle times for SRDF/AR are usually in the minutes to hours range. The RPO is double the cycle time in a worst case scenario. This may be a good fit for customers with relaxed RPOs.

The added benefit of having a longer cycle time is that the locality of reference will likely increase. This is because there is a much greater chance of a track being updated more than once in a one-hour interval than in, say, a 30-second interval. The increase in locality of reference shows up as reduced bandwidth requirements for the final solution.

Before SRDF/AR can be started, instantiation of the database has to have taken place. In other words, a full baseline copy of all the volumes that are going to participate in the SRDF/AR replication must be executed first. This means a full establish to the BCVs in the source array, a full SRDF establish of the BCV/R1s to the R2s, and a full establish of the R2s to the BCVs in the target array is required. There is an option to automate the initial setup of the relationship.

As with other SRDF solutions, SRDF/AR does not require a host at the DR site. The commands to update the R2s and manage the synchronization of the BCVs in the remote site are all managed in-band from the production site.
How to restart in the event of a disaster

In the event of a disaster, it is necessary to determine if the most current copy of the data is located on the remote site BCVs or R2s at the remote site. Depending on when in the replication cycle the disaster occurs, the most current version could be on either set of disks. A simple TimeFinder query can determine the status of the BCVs. If they are synchronized with the R2s, they should be split, and the R2s used for processing. If they are split from the R2s, they should be restored to the R2s, split, and then the R2s can be used for processing. If the status shows as in the process of synchronization, then the R2s should be used for processing.

SRDF/AR Multi-hop

SRDF Automated Replication Multi-hop, or SRDF/ AR Multi-hop, is an architecture that allows long distance replication with zero seconds of data loss through use of a bunker Symmetrix array. Production data is replicated synchronously to the bunker array, which is within 200 km of the production Symmetrix array allowing synchronous replication, but also far enough away that potential disasters at the primary site may not affect it. Typically, the bunker array is placed in a hardened computing facility.

BCVs in the bunker frame are periodically synchronized to the R2s and consistent split in the bunker frame to provide a dependent-write consistent point-in-time image of the data. These bunker BCVs also have an R1 personality, which means that SRDF in Adaptive Copy mode can be used to replicate the data from the bunker array to the target site. Since the BCVs are not changing, the replication can be completed in a finite length of time.

The length of time for the replication depends on:

- The size of the pipe (bandwidth) between the bunker location and the DR location
- The distance between the two locations
- The quantity of changed data
- The locality of reference of the changed data

On the remote Symmetrix array, another BCV copy of the data is made using the R2s. This is because the next SRDF/AR iteration replaces the R2 image, in a non-ordered fashion, and if a disaster
were to occur while the R2s were synchronizing, there would not be a valid copy of the data at the DR site. The BCV copy of the data in the remote Symmetrix array is commonly called the gold copy of the data. The whole process then repeats.

With SRDF/AR Multi-hop, there is minimal host impact. Writes are only acknowledged when they hit the cache of the bunker array and a positive acknowledgment is returned to the source Symmetrix array. Figure 43 shows this process.

The following steps describe how an SRDF/AR multi-hop environment processes:

1. BCVs are synchronized and consistently split against the R2s in the bunker Symmetrix array. The write activity is momentarily suspended on the source array to get a dependent-write consistent point-in-time image on the R2s in the bunker array, which creates a dependent-write consistent point-in-time copy of the data on the BCVs.

2. SRDF transmits the data on the bunker BCV/R1s to the R2s in the remote Symmetrix array.

3. When the BCV/R1 volumes are synchronized with the R2 volumes in the target Symmetrix array, the bunker BCV/R1s are established again with the R2s in the bunker array. This causes the SRDF links to be suspended between the bunker array and
the remote array. At the same time, an incremental establish is performed on the remote Symmetrix array to create a gold copy on the BCVs in that frame.

4. When the BCVs in the remote Symmetrix array are fully synchronized with the R2s, they are split, and the configuration is ready to begin another cycle.

The cycle repeats based on configuration parameters. The parameters can specify the cycles to begin at specific times, specific intervals, or to run immediately after the previous cycle completes.

It should be noted that even though cycle times for SRDF/AR multi-hop are usually in the minutes to hours range, the most current data is always in the bunker Symmetrix array. Unless there is a regional disaster that destroys both the primary site and the bunker site, the bunker array transmits all data to the remote DR site. This means zero data loss at the point of the beginning of the rolling disaster or an RPO of 0 seconds. This solution is a good fit for customers with a requirement of zero data loss and long distance DR.

An added benefit of having a longer cycle time means that the locality of reference will likely increase. This is because there is a much greater chance of a track being updated more than once in a one-hour interval than in, say, a 30-second interval. The increase in locality of reference manifests as reduced bandwidth requirements for the network segment between the bunker Symmetrix array and the DR array.

Before SRDF/AR can be started, initial instantiation of the DB2 system has to have taken place. In other words, a full baseline copy of all the volumes that are going to participate in the SRDF/AR replication must be executed first. This means a full establish of the R1s in the source location to the R2s in the bunker Symmetrix array. The R1s and R2s need to be synchronized continuously. Then a full establish from the R2s to the BCVs in the bunker array, a full SRDF establish of the BCV/R1s to the R2s in the DR array, and a full establish of the R2s to the BCVs in the DR array is performed. There is an option to automate this process of instantiation.

How to restart in the event of a disaster

In the event of a disaster, it is necessary to determine if the most current copy of the data is on the R2s on the remote site or on the BCV/R1s in the bunker Symmetrix array. Depending on when the
disaster occurs, the most current version could be on either set of disks. A simple TimeFinder query can determine the status of the BCVs. If they are synchronized with the R2s, then they should be split, and the R2s used for processing. If they are split from the R2s, they should be restored to the R2s, split, and then the R2s can be used for processing. If the status shows as in the process of synchronization, then the R2s should be used for processing.

**SRDF/Star**

The SRDF/Star disaster recovery solution provides advanced multi-site business continuity protection for enterprise environments. It combines the power of Symmetrix Remote Data Facility (SRDF) synchronous and asynchronous replication, enabling an advanced three-site business continuity solution. (See the SRDF/Star for z/OS User Guide).

SRDF/Star enables concurrent SRDF/S and SRDF/A operations from the same source volumes with the ability to incrementally establish an SRDF/A session between the two remote sites in the event of a primary site outage.

This software provides the ability to quickly reestablish protection between the two remote sites in the event of a primary site failure, and then, just as quickly, restore the primary site when conditions permit.

With SRDF/Star, enterprises can quickly resynchronize the SRDF/S and SRDF/A copies by replicating only the differences between the sessions, allowing for a fast resumption of protected services after a source site failure.

Figure 44 on page 223 is an example of an SRDF/Star configuration.
Enginuity 5874 and later support a feature called SRDF/Extended Distance Protection (SRDF/EDP). This feature allows a much more optimized and efficient structure to a three-site SRDF topology in that it allows the intermediate site (site B, in an A to B to C topology) to have a new device type known as a diskless R21. This arrangement requires that the secondary system (site B) to be at Enginuity 5874 or later (and can also be a VMAX array), while sites A and C can be running either DMX-4 with Enginuity 5773 or a Symmetrix VMAX array with Enginuity 5874 or later. Thus, SRDF/EDP allows replication between the primary site A and tertiary site B without the need for SRDF BCVs or any physical storage for the replication at the secondary site B. Figure 45 on page 224 depicts an SRDF/EDP configuration.
The diskless R21 differs from the real disk R21, introduced in an earlier release of Enginuity, which had three mirrors, one at each of the three sites. The diskless R21 device is a new type of device which does not have any local mirrors. Furthermore, it has no local disk space allocated on which to store the user data, and therefore, reduces the cost of disk storage in the site B system. This results in only two full copies of data, one on the source site A and one on the target site C.

The purpose of a diskless R21 device is to cascade data directly to the remote R2 disk device. When using a diskless R21 device, the changed tracks received from the R1 mirror are saved in cache until these tracks are sent to the R2 disk device. Once the data is sent to the R2 device, and the receipt is acknowledged, the cache slot is freed, and the data no longer exists on the R21 Symmetrix array.

This advantageous approach to three-site SRDF means that a customer only needs a Symmetrix subsystem with vault and SFS drives plus enough cache to hold common area, user data/updates (customer data), and device tables, thereby reducing the overall solution cost. It highlights a serious attempt to address a greener alternative to the device sprawl brought about by multi-site business continuity requirements and is sure to be welcomed by many customers deploying three-site DR solutions. The R21 diskless device still uses the device table, like a disk device, and also consumes a Symmetrix device number. They are not addressable by a host or assigned to a DA and therefore cannot be accessed for any I/Os.

Other considerations with diskless R21 devices include:

- They can only be supported on GigE and Fibre Channel directors.
- They cannot participate in dynamic sparing since the DA microcode blocks any type of sparing and performing iVTOC against them.
- They cannot be in an SRDF pair with other diskless devices.
- When used for SRDF/A operations, all devices in the SRDF/A session must be diskless. Non-diskless device types are not allowed.
- All Symmetrix replication technologies other than SRDF (TimeFinder/Mirror, Snap, and Clone) do not function with diskless devices configured as either the source or the target of the intended operation.
- Symmetrix Data Differential Facility (SDDF) sessions are allowed on diskless devices.

**DB2 data sharing restart considerations**

When restarting a remotely replicated DB2 data sharing environment, additional tasks are required to ensure a successful restart. This section discusses issues to consider when performing this type of restart.

**Group restart**

Data sharing restart requires that a group restart be performed. All members of the group must be restarted. DB2 V7.1 introduced a new feature, restart light. The `LIGHT(YES)` parameter can be used on the `START DB2` command to restart a DB2 member in light mode. This causes that member to recover the retained locks, then terminates the member normally after forward and backward recovery is complete. No new work is accepted.

Restart light mode is intended only for a cross-system restart to a system that may not have adequate capacity to maintain all DB2 and IRLM tasks for a data sharing group. A DB2 system started with the light option is not registered with the Automatic Resource Manager (ARM) and will not, therefore, be automatically restarted.

*Note:* Refer to IBM’s *DB2 10 for z/OS Data Sharing: Planning and Administration* for a full description of the restart light feature.
Coupling facility structures

Any existing DB2 data sharing structures must be cleaned up prior to restart by means of `SETXCF` commands. The following are examples of the `SETXCF` commands:

```
SETXCF FORCE, CONNECTION, STRNAME=DBG1_LOCK1, CONNAME=ALL
SETXCF FORCE, CONNECTION, STRNAME=DBG1_GBP0, CONNAME=ALL
SETXCF FORCE, CONNECTION, STRNAME=DBG1_GBP1, CONNAME=ALL
SETXCF FORCE, CONNECTION, STRNAME=DBG1_SCA, CONNAME=ALL
SETXCF FORCE, STRUCTURE, STRNAME=DBG1_LOCK1
SETXCF FORCE, STRUCTURE, STRNAME=DBG1_GBP0
SETXCF FORCE, STRUCTURE, STRNAME=DBG1_GBP1
SETXCF FORCE, STRUCTURE, STRNAME=DBG1_SCA
```

Other restart considerations

When restarting a data sharing group after a failure, it is probable that LPL entries will be created by DB2. LPLs caused by retained data sharing locks during `GROUP RESTART` are common, but should be resolved after issuing the following commands:

```
-DIS DB(*) SPACE(*) RESTRICT
```

Once this information is obtained, then issue the following command for each database/table space listed in the previous command:

```
-STA DB(xxxxxxxx) SPACE(xxxxxxxx)
```

This procedure triggers the DB2 LPL recovery process.

**Note:** If there are any entries in the restricted list, DSNDB01 should be started first and DSNDB06 should be started second.
High-availability solutions

Customers that cannot tolerate any data loss or outages, whether planned or unplanned, require high-availability solutions to maintain business continuity. EMC solutions that provide high-availability are discussed in this section.

AutoSwap

EMC AutoSwap is a software package that allows I/O to be redirected from a primary (R1) device in a Synchronous SRDF relationship to its partner (R2) device with minimal impact to a running application. AutoSwap allows users to manage planned and unplanned events.

ConGroup contains a license-key enabled AutoSwap extension that handles automatic workload swaps between Symmetrix systems when the AutoSwap software detects an unplanned outage or problem.

AutoSwap can be used in both shared and non-shared DASD environments and uses standard z/OS operating system services to ensure serialization and to affect swaps. AutoSwap uses the Cross System Communication facility (CSC) of SCF to coordinate swaps across multiple z/OS images in a shared DASD or parallel sysplex environment.

Geographically Dispersed Disaster Restart

Geographically Dispersed Disaster Restart (GDDR) is a mainframe software product that standardizes and automates business recovery for both planned and unplanned outages. GDDR is designed to restart business operations following disasters ranging from the loss of computer capacity and/or disk array access, to total loss of a single data center, or a regional disaster including the loss of dual data centers. GDDR achieves this goal by providing the automation quality controls to the functionality of many EMC and third-party hardware and software products required for business restart. GDDR can automate business recovery for two-site or three-site configurations.
The EMC GDDR complex

Because GDDR is used to restart following disasters, it does not reside on the same hosts that it is seeking to protect. For this reason, GDDR resides on separate logical partitions (LPARs) from the hosts that run the customer’s application workloads. Thus, in a three-site SRDF/STAR configuration, GDDR is installed on a control LPAR at each site. Each GDDR node is aware of the other two GDDR nodes by means of network connections between each site. This awareness is required to detect disasters, identify survivors, nominate the leader, and then take the necessary actions to recover the customer’s business at one of the customer-chosen surviving sites.

To achieve the task of business restart, GDDR automation extends well beyond the disk layer, where EMC has traditionally focused, and into the host operating system layer. It is at this layer that sufficient controls and access to third-party software and hardware products exist to enable EMC to provide automated recovery services.

Figure 46 on page 229 shows a configuration managed by GDDR with SRDF/Star and AutoSwap:
Understanding Disaster Restart and Disaster Recovery

Figure 46  GDDR with SRDF/Star and AutoSwap

In Figure 46 on page 229, DC1 is the primary site, where the production workload is located. DC1 is also the primary DASD site, where the R1 DASD is located. DC2 is the secondary site and contains the secondary R2 DASD.

Sites DC1 and DC2 are the primary and secondary data centers of critical production applications and data. They are considered fully equivalent for strategic production applications, connected with highly redundant direct network links. At all times, all production data is replicated synchronously between the two sites.

Site DC3 is the tertiary data center for critical production applications and data. It is connected with redundant network to both DC1 and DC2. Data is replicated asynchronously from the current primary DASD site with an intended recovery point objective (RPO) in a short number of minutes.
GDDR and business continuity

Implementing SRDF/S and/or SRDF/A, TimeFinder and other EMC and non-EMC technologies are necessary prerequisites to be able to recover or restart following some level of interruption or disaster to normal IT operations. However, these are foundation technologies. Because of the complexity of operation of these various products, including known or unknown dependencies and sequencing of operations, it is necessary to deploy automation to coordinate and control recovery and restart operations.

Figure 47 shows EMC’s foundation technologies used in a GDDR environment.

Figure 47  EMC foundation products for GDDR

For more information about GDDR, see EMC Geographically Dispersed Disaster Restart Concepts and Facilities and EMC Geographically Dispersed Disaster Restart Product Guide.
This chapter presents these topics:

- **Overview** .......................................................... 232
- **The performance stack** ........................................... 232
- **Storage system layer considerations** ..................... 234
- **Traditional DB2 layout recommendations** ............. 234
- **Symmetrix system performance guidelines** ............ 236
- **RAID considerations** ............................................ 245
- **Flash drives** .......................................................... 249
- **Data placement considerations** ............................ 254
- **Other layout considerations** ................................. 259
- **VSAM striping for DB2 active logs** ......................... 262
- **DB2 V9 and DFSORT** ........................................... 262
- **Remote replication performance considerations** ...... 263
- **DB2 use of PAVs on Symmetrix arrays** .................. 265
- **Extended address volumes** .................................... 266
Overview

Monitoring and managing database performance should be a continuous process in all DB2 environments. Establishing baselines and then collecting database performance statistics to compare against them are important to monitor performance trends and maintain a smoothly running system. This chapter discusses the performance stack and how database performance should be managed in general. Subsequent sections, discuss Symmetrix-specific layout and configuration issues to help ensure the database meets the required performance levels.

The performance stack

Performance tuning involves the identification and elimination of bottlenecks in the various resources that make up the system. Resources include the application, the code (SQL) that drives the application, the database, the host, and the storage. Tuning performance involves analyzing each of these individual components that make up an application, identifying bottlenecks or potential optimizations that can be made to improve performance, implementing changes that eliminate the bottlenecks or improve performance, and verifying that the change has improved overall performance. This is an iterative process and is performed until the potential benefits from continued tuning are outweighed by the effort required to tune the system.

Figure 48 on page 233 shows the various layers that need to be examined as a part of any performance analysis. The potential benefits achieved by analyzing and tuning a particular layer of the performance stack are not equal, however. In general, tuning the upper layers of the performance stack, that is the application and SQL statements, provide a much better ROI than tuning the lower layers, such as the host or storage layers. For example, implementing a new index on a heavily used table that changes logical access from a full table scan to index lookup with individual row selection can vastly improve database performance if the statement is run many times (thousands or millions) per day.

When tuning a DB2 system, developers, DBAs, systems administrators, and storage administrators need to work together to monitor and manage the process. Efforts should begin at the top of the stack, and address application and SQL statement tuning before...
The primary goal to achieve at all levels of the performance stack is disk I/O avoidance. In theory, an ideal database environment is one in which most I/Os are satisfied from memory rather than going to disk to retrieve the required data. In practice, however, this is not realistic. Careful consideration of the disk I/O system is necessary. Optimizing performance of a DB2 system on an EMC Symmetrix array involves a detailed evaluation of the I/O requirements of the proposed application or environment. A thorough understanding of the performance characteristics and best practices of Symmetrix arrays, including the underlying storage components (disks, directors, and so on), is also needed. Additionally, knowledge of complementary software products, such as EMC SRDF, TimeFinder, Symmetrix Optimizer, and backup software, along with how utilizing these products affects the database, is important for maximizing
performance. Ensuring optimal configuration for a DB2 system requires a holistic approach to application, host, and storage configuration planning. Configuration considerations for host- and application-specific parameters are beyond the scope of this TechBook.

Storage system layer considerations

What is the best way to configure DB2 on EMC Symmetrix storage? This is a frequently asked question from customers. However, before recommendations can be made, a detailed understanding of the configuration and requirements for the database, host, and storage environment is required. The principal goal for optimizing any layout on a Symmetrix subsystem is to maximize the spread of I/O across the components of the array, reducing or eliminating any potential bottlenecks in the system. The following sections examine the trade-offs between optimizing storage performance and manageability for DB2. They also contain recommendations for laying out DB2 for z/OS on EMC Symmetrix arrays.

Traditional DB2 layout recommendations

Traditional best practices for DB2 system layouts focus on avoiding contention between storage-related resources. Eliminating contention involves understanding how the database manages the data flow process and ensures that concurrent or near-concurrent storage resource requests are separated on to different physical spindles. Many of these recommendations still have value in a Symmetrix environment. Before examining other storage-based optimizations, a brief digression to discuss these recommendations is made.

General principles for layout

Some of the axioms that have guided database layouts for performance in the past have now grown obsolete. Among these are:

- Separate active logs from table space data
- Separate index components from data components
- Isolate TEMPORARY table spaces from DATA and INDEX information
While outmoded, these guidelines still hold value and in an ideal world, would still be used. The challenge that faces DBAs today is that the physical disks that support the database are now so large that any form of isolation is going to be wasteful of space. In addition, in modern sophisticated storage array implementations, it is laborious and time-consuming to determine the physical composition of a volume presented to a host. For these reasons, the following general guidelines best suit DB2 for z/OS deployments on Symmetrix subsystems:

- Spread the data across as many disks as possible
- Spread the temp space and logs across multiple disks
- Separate archive logs from active logs

The key point of any recommendation for optimizing the layout is that it is critical to understand both the type (sequential or random), size (small or large), and quantity (low, medium, or high) of I/O against the various table spaces and other elements (logs, temp space, and so on) of the database. Without clearly understanding data elements and the access patterns expected against them, serious contention issues on the back-end directors or physical spindles may arise that can negatively impact DB2 performance. Knowledge of the application, both data elements and access patterns, is critical to ensuring high performance in the database environment.

### DB2 layouts and replication considerations

When planning to use array replication, it is prudent to organize DB2 table spaces in such a way to facilitate recovery. Since array replication techniques copy volumes at the volume level (as seen by the host), all table spaces should be created on a set of disks dedicated to the DB2 system and should not be shared with other applications or DB2 systems.

In addition to isolating the database to be copied onto its own dedicated volumes, the DB2 datasets should also be divided into two parts: The data structures and the recovery structures. The recovery structures comprise the bootstrap datasets (BSDS), active logs, and archive logs. The data volumes hold the DB2 directory, DB2 catalog and application table spaces. This division allows the two parts to be manipulated independently if a recovery becomes necessary.
With any kind of volume-based replication, the ICF catalogs that reference the datasets that are being restored should also be restored. In this case, there should be two catalogs, one for the recovery structures (residing on one of the recovery structure volumes) and one for the data structures (residing on one of the data structure volumes).

Symmetrix system performance guidelines

Optimizing performance for DB2 for z/OS in a Symmetrix environment is very similar to optimizing performance for all applications on the storage array. To maximize performance, a clear understanding of the I/O requirements of the applications accessing storage is required. The overall goal when laying out an application on disk devices in the back end of the Symmetrix controller is to reduce or eliminate bottlenecks in the storage system by spreading out the I/O across all of the array's resources. Inside a Symmetrix array, there are a number of resources to consider:

- Front-end connections into the Symmetrix array. These includes the number of connections from the host to the array that are required, and whether front-end FICON ports are directly connected or connected through a FICON director.
- Memory cache in the Symmetrix controller. All host I/Os pass through cache on the array. I/O can be adversely affected if insufficient cache is configured in the array for the environment. Also, writes to individual hypervolumes, or to the array as a whole, may be throttled when a threshold known as the write-pending limit (discussion to follow) is reached.
- Back-end considerations. There are two sources of possible contention in the back-end of the Symmetrix array: The back-end directors and the physical spindles. Proper layout of the data on the disks is needed to ensure satisfactory performance.

Front-end connectivity

Optimizing front-end connectivity requires an understanding of the number and size of I/Os (both reads and writes) that are sent between the hosts and the Symmetrix subsystem. There are limitations to the quantity of I/O that each front-end director port, each front-end director processor, and each front-end director board
are capable of handling. Additionally, FICON director fan-out counts, that is the number of hosts that can be attached through a FICON director to a single front-end port, need to be carefully managed.

A key concern when optimizing front-end performance is determining which of the following I/O characteristics is more important in the customer’s environment:

- Input/output operations per second (IOPS)
- Throughput (MB/s)
- A combination of IOPS and throughput

In OLTP database applications, where I/Os are typically small and random, IOPS is the more important factor. In decision support systems (DSS), where transactions generally require large sequential table or index scans, throughput is the more critical factor. In some databases, a combination of OLTP and DSS-like I/Os are required. Optimizing performance in each type of environment requires tuning the host I/O size.

**Figure 49 on page 238** depicts the relationships between the page size of a random read request from the host and both IOPS and throughput needed to fulfill that request from the Symmetrix array. It shows that the maximum number of IOPS is achieved using smaller page sizes, such as 4 KB (4096). For OLTP applications, where the typical DB2 page size is 4 KB, it shows that the Symmetrix subsystem provides higher IOPS, but decreased throughput. The opposite is also true for DSS applications. Tuning the database to send larger I/O sizes for DSS applications can increase the overall throughput (that is, MB/s) from the front-end directors on the Symmetrix controller. Database page sizes are generally larger (16 KB or even 32 KB) for DSS applications.

Currently, each FICON port on the Symmetrix subsystem is theoretically capable of 400 MB/s of throughput. In practice, however, the throughput that is available per port is significantly less and depends on the I/O size and on the shared utilization of the port and processor on the director. Increasing the size of the I/O from the host perspective decreases the number of IOPS that can be performed, but increases the overall throughput (MB/s) of the port. As a result, increasing the I/O page size on the host is beneficial for overall performance in a DSS environment. Limiting total throughput to a fraction of the theoretical maximum (70 percent is probably a good guideline) ensures that enough bandwidth is available for connectivity between the Symmetrix subsystem and the host.
Symmetrix cache

The Symmetrix cache plays a key role in improving I/O performance in the storage system. Cache improves performance by allowing write acknowledgements to be returned to a host when data is received in cache in the storage controller, rather than being fully destaged to the physical disk drives. Additionally, read operations benefit from cache when sequential requests from the host allow follow-on reads to be prestaged in cache. Symmetrix cache should be considered an extension to the DB2 buffer pool.

The next sections briefly describe how the Symmetrix cache is used for writes and reads, and then discuss performance considerations for it.

Write operations and the Symmetrix cache

All write operations on a Symmetrix array are serviced by cache. When a write is received by the front-end director, a cache slot is found to service the write operation. Since cache slots represent the underlying hypervolume, if a prior read or write operation caused the required data to already be loaded into cache, the existing cache slot may be used to store the write data. If a cache slot that represents...
the storage area was not found, a call is made to locate a free cache slot for the write. The write operation is moved to the cache slot and the slot is then marked write pending and normal ending status is returned to the host. At some later point, Enginuity destages the write to physical disk. Overall system load, physical disk activity read operations from the physical disk, and availability of cache determine when to destage.

Cache is used to service the write operation to optimize the performance of the host system. Because write operations to cache are significantly faster than physical writes to disk media, the write is reported as complete to the host operating system much earlier. Mirrored cache memory, cache vaulting, battery backup, and priority destage functions within the Symmetrix system ensure that no data loss occurs in the event of system power failure.

If higher priority operations (read activity is one such operation) delay the write operation to a given disk, the write-pending slot remains in cache longer. Cache slots are allocated as needed to a volume for this purpose. Enginuity calculates thresholds for allocations to limit the saturation of cache by a single hypervolume. These limits are referred to as write-pending limits.

Cache allocations are based on a per-hypervolume basis. As write-pending thresholds are reached, additional allocations may occur, as well as reprioritization of write activity. As a result, write operations to the physical disks may increase in priority to ensure that excessive cache allocations do not occur. This is discussed in more detail in later sections.

Cache enables buffering of writes and allows for a steady stream of write activity to service the destaging of write operations from a host. In a bursty write environment, this serves to even out the write activity. If the write activity constantly exceeds the low-write priority to the physical disk, Enginuity raises the priority of write operations to attempt to meet the write demand. Ultimately, if write workload from the host exceed the physical disk ability to write, the volume maximum write-pending limit may be reached. In this condition, new cache slots are only allocated for writes to a particular volume once a currently allocated slot is freed by destaging it to disk. This condition, if reached, may severely impact write operations to a single hypervolume.
Read operations and the Symmetrix cache

As discussed in the previous section, read operations typically have an elevated priority for service from the physical disks. As user processes commonly need to wait for an I/O operation to complete before continuing, this is generally a good practice for storage arrays, especially those able to satisfy write operations from cache.

When a read request is detected from a host system, Enginuity sees if a corresponding cache slot containing the storage area exists. If so, the read request may be serviced immediately (this is considered a read hit). If the required data is not in cache, and free slots are available, the read operation must wait for a transfer from disk (this is called a short read miss). If no cache slot exists for the read to be transferred into, then a cache slot must be allocated and the read physically transferred into the slot (this is referred to as a long read miss).

Although cache slots themselves are 64 KB (32 KB prior to DMX-3), a cache slot may contain only the requested data. That is, if a read request is made for an 8 KB block, then only that 8 KB block is transferred into the cache slot, as opposed to reading the entire 64 KB track from disk. The smallest read request unit is 8 KB.

Symmetrix cache performance considerations

An important performance consideration is to ensure that an appropriate amount of cache is installed in the array. All I/O requests from hosts attached to the array are serviced from the Symmetrix cache. Symmetrix cache can be thought of as an extension to database buffering mechanisms. Therefore, many database application environments can benefit from additional Symmetrix cache. With newly purchased arrays, appropriately sizing the cache is performed by the EMC sales team, based upon the number and size of physical spindles, configuration (including number and type of volumes), replication requirements (SRDF, for example), and customer requirements.

Cache usage can be monitored through a number of Symmetrix monitoring tools. Primary among these is Ionix® ControlCenter Performance Manager. Performance Manager contains a number of views that analyze Symmetrix cache utilization at both the hypervolume and overall system level. Views provide detailed information on specific component utilizations, including disks, directors (front-end and back-end), and cache utilization.
Symmetrix cache plays a key role in host I/O read and write performance. Read performance can be improved through prefetching by the Symmetrix controller if the reads are sequential in nature. Enginuity algorithms detect sequential read activity and prestage reads from disk in cache before the data is requested. Write performance is greatly enhanced because all writes are acknowledged back to the host when they reach Symmetrix cache, rather than when they are written to disk. While reads from a specific hypervolume can use as much cache as is required to satisfy host requests, assuming free cache slots are available, there are limits to the number of writes that can be written to a single volume (that is, the write-pending limit discussed previously). Understanding the Enginuity write-pending limits is important when planning for optimal performance.

As previously discussed, the write-pending limit is used to prevent high write rates to a single hypervolume from consuming all of the storage array cache for its use, at the expense of performance for reads or writes to other volumes. The write-pending limit for each hypervolume is determined at storage system startup. It depends on the number and type of volumes configured, and the amount of cache available. The limit does not depend on the actual size of each volume. The more cache available, the more write requests that can be serviced in cache by each individual volume. If the maximum write-pending limit for a volume is reached, destaging to disk must take place before new writes can come in. This forced destaging to disk before a new write can be received into cache limits writes to that particular volume to physical disk write speeds. Forced destage of writes can significantly reduce performance to a hypervolume if the write-pending limit is reached. If performance problems with a particular volume are identified, an initial step in determining the source of the problem should include verifying the number of writes and the write-pending limit for that volume.

In addition to limits imposed at the hypervolume level, there are additional write-pending limits imposed at the system level. Two key cache utilization points for the Symmetrix controller are reached when 50 percent and 80 percent of the cache is used for pending writes. Under normal operating conditions, satisfying read requests from a host has greater priority than satisfying write requests. However, when pending writes consume 50 percent of cache, the Symmetrix array then prioritizes reads and writes equally. This reprioritization can have a profound affect on database performance. The impact is even more pronounced if cache utilization for writes
reaches 80 percent. At that point, the array begins a forced destage of writes to disk, with discernible performance degradation of both writes and reads. If this threshold is reached, it is a clear indicator that both the cache and the total I/O on the array need to be reexamined.

**Dynamic Cache Partitioning**

Enginuity 5772 provides for the capability to divide the Symmetrix cache into partitions based on user-defined parameters. This allows for provision of cache to competing workloads in order that the higher priority workloads do not get impacted by cache consumption of lower priority activities.

Dynamic Cache Partitioning can also be used to fence off cache usage of Flash drives (available in DMX-4 and VMAX arrays) since these drives are already serving I/O at memory type speeds.

**Related ZPARM settings**

The two ZPARM settings that relate to storage controller cache are SEQCACH and SEQPRES. These settings were introduced to stop pollution of the controller cache for certain DB2 operations. In a Symmetrix configuration, these settings are ignored.

**Back-end considerations**

For the most part, back-end considerations are eliminated when using Virtual Provisioning. The wide striping provided by the thin pools eliminates hot spots on the disks and thus allows utilization levels to be driven much higher. For more details on Virtual Provisioning see “Advanced Storage Provisioning” on page 67. The rest of this section on back-end considerations apply to thick implementations.

Back-end considerations are typically the most important part of optimizing performance on the Symmetrix array. Not withstanding the DMX-4 and VMAX support of enterprise Flash drives, advances in disk technologies have not kept up with performance increases in other parts of the storage array, such as director and bandwidth (that is, Direct Matrix versus bus) performance. Disk access speeds have increased by a factor of three to seven in the last decade, while other components have easily increased one to three orders of magnitude. Consequently, most performance bottlenecks in the Symmetrix controller are attributable to physical spindle limitations.
An important consideration for back-end performance is the number of physical spindles available to handle the anticipated I/O load. Each disk is capable of a limited number of operations. Algorithms in the Symmetrix Enginuity operating environment optimize I/Os to the disks. Although this helps to reduce the number of reads and writes to disk, particularly for random reads, is still a limiting factor. If an insufficient number of physical disks are available to handle the anticipated I/O workload, performance can suffer. It is critical to determine the number of spindles required for a DB2 for z/OS system implementation based on I/O performance requirements, and not solely on the physical space considerations.

To reduce or eliminate back-end performance issues on the Symmetrix subsystem, care should be taken to spread access to the disks across as many back-end directors and physical spindles as possible. This means that performance is improved by spreading data across the back-end directors and disks, rather than allocating specific applications to specific physical spindles. Significant attention should be given to balancing the I/O on the physical spindles.

Implementing Symmetrix Optimizer may also help to reduce I/O contention between hypervolumes on a physical spindle. Symmetrix Optimizer identifies I/O contention on individual hypervolumes and nondisruptively moves one of the hypervolumes to a new location on another, lightly used, disk. Symmetrix Optimizer is an invaluable tool in helping to reduce contention on physical spindles should workload requirements change in an environment.

Placement of data on the disks is another performance consideration. Because of the rotational properties of disk platters, tracks on the outer parts of the disk perform better than inner tracks. While the Symmetrix Enginuity algorithms smooth out much of this variation, small performance increases can be achieved by placing high I/O objects on the outer parts of the disk. Of more importance, however, is minimizing the seek times associated with the disk head moving between hypervolumes on a spindle. Physically locating higher I/O devices together on the disks can significantly improve performance. Disk head movement across the platters (seek time) is a large source of latency in I/O performance. Placing higher I/O devices contiguously may reduce disk head movement, increasing I/O performance of that physical spindle.
Host addressing limitations

There are also limitations on the number of UCBs that can be addressed in a single sysplex. Currently the number is around 65,000. While this number may seem high, it can easily be reached within a large sysplex environment using PAVs.

Consequently, it might be prudent to design a storage configuration that includes more of the larger mod sizes (MOD-27, MOD-54) and less of the smaller sizes (MOD-1, MOD-3 and MOD-9).

Configuration recommendations

The key recommendations for configuring the Symmetrix array for optimal performance include the following:

- Understand the I/O—It is critical to understand the characteristics of the database I/O, including the number, type (read or write) size, location (that is, table space datasets, logs), and the sequentiality of the I/Os. Empirical data or estimates are needed to assist in planning.

- Physical spindles—The number of disk drives in the Symmetrix array should first be determined by calculating the number of I/Os required, rather than solely based on the physical space needs. The key is to ensure that the front-end needs of the applications can be satisfied by the flow of data from the back end.

- Spread the I/O—Both reads and writes should be spread across the physical resources (front-end and back-end ports, physical spindles, hypervolumes) of the Symmetrix controller. This helps to prevent bottlenecks, such as hitting port or spindle I/O limits, or reaching write-pending limits on a hypervolume.

- Bandwidth—A key consideration when configuring connectivity between a host and the Symmetrix subsystem is the expected bandwidth required to support database activity. This requires an understanding of the size and number of I/Os between the host and relationship between host block size and IOPS. Connectivity considerations for both the number of FICON channels and Symmetrix front-end ports is required.
RAID considerations

Integrated Cached Storage Arrays provide multiple ways to protect and manage database data. The options that are chosen at the array level can affect the operation of the running database. The choice of RAID 1 and RAID 5, RAID 6, or RAID 10 may determine in part the how the DB2 system performs as each of these protection mechanisms has a different performance profile. The following sections provide a brief description of RAID configurations provided by the Symmetrix array.

Types of RAID

The following list defines RAID configurations that are available on the Symmetrix array:

◆ Unprotected—This configuration is not typically used in a Symmetrix environment for production volumes. BCVs and occasionally R2 devices (used as target devices for SRDF) can be configured as unprotected volumes.

◆ RAID 1—These are mirrored devices and are the most common RAID type in a Symmetrix subsystem. Mirrored devices require writes to both physical spindles. However, intelligent algorithms in the Enginuity operating environment can use both copies of the data to satisfy read requests that are not already in the cache of the array. RAID 1 offers optimal availability and performance and is more costly than other RAID protection options since 50 percent of the disk space is used for protection purposes.

◆ RAID 10—These are striped and mirrored devices. In this configuration, Enginuity stripes data of a logical device across several physical drives. Four Symmetrix devices (each a fourth the size of the original mainframe device) appear as one mainframe device to the host, accessible through one channel address (more addresses can be provided with PAVs). Any four devices can be chosen to define a group provided they are equally sized, of the same type (3380, 3390, and so on), and have the same mirror configuration. Striping occurs across this group of four devices with a striping unit of one cylinder.

◆ RAID 6—Protection schemes, such as RAID 1 and RAID 5, can protect a system from a single physical drive failure within a mirrored pair or RAID group. RAID 6 supports the ability to
rebuild data in the event that two drives fail within a RAID group. RAID 6 on the Symmetrix array comes in two configurations, (6+2) and (14+2).

EMC’s implementation of RAID 6 calculates two types of parity, which is key in order for data to be reconstructed following a double drive failure. Horizontal parity is identical to RAID 5 parity, which is calculated from the data across all the disks. Diagonal parity is calculated on a diagonal subset of data members.

RAID 6 provides high data availability but could be subject to significant write performance impact due to horizontal and diagonal parity generation, as with any RAID 6 implementation. Therefore, RAID 6 is generally not recommended for write-intensive workloads.

Permanent sparing can also be used to further protect RAID 6 volumes. RAID 6 volumes also benefit from a new data structure called the Physical Consistency Table that allows them to occupy only a single mirror position.

- **RAID 5**—This protection scheme stripes parity information across all volumes in the RAID group. RAID 5 offers good performance and availability, at a decreased cost. Data is striped using a stripe width of four tracks. RAID 5 is configured either as RAID 5 3+1 (75 percent usable) or RAID 5 7+1 (87.5 percent usable) configurations. Figure 50 on page 247 shows the configuration for RAID 5 3+1. Figure 51 on page 247 shows how a random write in a RAID 5 environment is performed.
The following describes the process of a random write to a RAID 5 volume:

1. A random write is received from the host and is placed into a data slot in cache to be destaged to disk.

2. The write is destaged from cache to the physical spindle. When received, parity information is calculated in cache on the drive by reading the old data and using an exclusive-or calculation with the new data.

3. The new parity information is written back to Symmetrix cache.

4. The new parity information is written to the appropriate parity location on another physical spindle.
The availability and performance requirements of the applications that utilize the Symmetrix subsystem determine the appropriate level of RAID to configure in an environment. Combinations of RAID types are configurable in the Symmetrix array with some exceptions. For example, storage may be configured as a combination of RAID 1 and 3+1 RAID 5 devices. Combinations of 3+1 and 7+1 RAID 5 are allowed in the same Symmetrix array when using 5772 Enginuity or later.

Until recently, RAID 1 was the predominant choice for RAID protection in Symmetrix storage environments. RAID 1 provides maximum availability and enhanced performance over other available RAID protections. In addition, performance optimizations such as Symmetrix Optimizer, which reduces contention on the physical spindles by nondisruptively migrating hypervolumes, and Dynamic Mirror Service Policy, which improves read performance by optimizing reads from both mirrors, were only available with mirrored volumes. While mirrored storage is still the recommended choice for RAID configurations in the Symmetrix array, the addition of RAID 5 storage protection provides customers with a reliable, economical alternative for their production storage needs.

RAID 5 storage protection became available with the 5670+ release of the Enginuity operating environment. RAID 5 implements the standard data striping and rotating parity across all members of the RAID group (either 3+1 or 7+1). Additionally, Symmetrix Optimizer functionality is available with RAID 5 to reduce spindle contention. RAID 5 provides customers with a flexible data protection option for dealing with varying workloads and service-level requirements.

RAID recommendations

While EMC recommends RAID 1 or RAID 10 to be the primary choice in RAID configuration for reasons of performance and availability, DB2 systems can be deployed on RAID 5 protected disks for all but the highest I/O performance-intensive applications. Databases used for test, development, QA, or reporting are likely candidates for using RAID 5 protected volumes.

Another potential candidate for deployment on RAID 5 storage is DSS applications. In many DSS environments, read performance greatly outweighs the need for rapid writes. This is because DSS applications typically perform loads off-hours or infrequently (once per week or month). Read performance in the form of database user
queries, is significantly more important. Since there is no RAID penalty for RAID 5 read performance (only write performance) these types of applications are generally good candidates for RAID 5 storage deployments. Conversely, production OLTP applications typically require small random writes to the database, and therefore, are generally more suited to RAID 1 or RAID 10 storage.

An important consideration when deploying RAID 5 is disk failures. When a disk containing RAID 5 members fails, two primary issues arise, performance and data availability. Performance is affected when the RAID group operates in the degraded mode because the missing data must be reconstructed using parity and data information from other members in the RAID group. Performance is also affected when the disk rebuild process is initiated after the failed drive is replaced or a hot spare is disk is activated. Potential data loss is the other important consideration when using RAID 5. Multiple drive failures that cause the loss of multiple members of a single RAID group result in loss of data. While the probability of such an event is infinitesimal, the potential in 7+1 RAID 5 environment is much higher than that for RAID 1. Consequently, the probability of data loss, because of the loss of multiple members of RAID 5 group, should be carefully weighed against the benefits of using RAID 5.

The bottom line in choosing a RAID type is ensuring that the configuration meets the needs of the customer’s environment. Considerations include read/write performance, balancing the I/O across the spindles and back end of the Symmetrix controller, tolerance for reduced application performance when a drive fails, and the consequences of losing data in the event of multiple disk failures. In general, EMC recommends RAID 10 for all types of DB2 systems. RAID 10 stripes data across multiple devices and should be used for all large MOD sizes.

RAID 5 configurations may be beneficial for many low I/O rate applications.

EMC has enhanced Symmetrix Engineuity 5773 and 587x to integrate enterprise-class Flash drives directly into the DMX-4 and VMAX storage arrays. With this capability, EMC creates a Tier 0 ultra-performance storage tier that transcends the limitations previously imposed by magnetic disk drives. By combining enterprise-class Flash drives optimized with EMC technology and
advanced Symmetrix functionality (including Virtual LUN migration and FAST) organizations now have new tiering options previously unavailable from any vendor.

Flash drives provide maximum performance for latency-sensitive applications. Flash drives, also referred to as solid-state drives (SSD), contain no moving parts and appear as normal Fibre Channel drives to existing Symmetrix management tools, allowing administrators to manage Tier 0 storage without special processes or custom tools. Tier 0 Flash storage is ideally suited for applications with high transaction rates and those requiring the fastest possible retrieval and storage of data, such as currency exchange and electronic trading systems, real-time data feed processing. A Symmetrix subsystem with Flash drives can deliver single-millisecond application response times and up to 30 times more IOPS than traditional 15,000 rpm Fibre Channel disk drives. Additionally, because there are no mechanical components, Flash drives require up to 98 percent less energy per IOPS than traditional disk drives.

Magnetic disk history

For years, the most demanding enterprise applications have been limited by the performance of magnetic disk media. Tier 1 performance in storage arrays has been unable to surpass the physical limitations of hard disk drives. With EMC’s addition of Flash drives to DMX-4 and VMAX arrays, organizations can now take advantage of ultra-high performance optimized for the highest-level enterprise requirements. Flash drives for Tier 0 requirements deliver unprecedented performance and response times for DMX-4 and VMAX arrays, far surpassing that of any other product on the market today. Figure 52 on page 251 shows, using a logarithmic scale, how magnetic disk capacity and performance have grown over the last decade and a half as compared to the demand of applications.
It is important to understand that any I/O request from the host is serviced by the Symmetrix array from its global cache. Under normal circumstances, a write request is always written to cache and incurs no delay due to physical disk access. In the case of a read request, if the requested data is in the global cache either because of recent read or write, or due to sequential prefetch, the request is immediately serviced without disk I/O. A read serviced from cache without causing disk access is called a read hit. If the requested data is not in the global cache, the Symmetrix subsystem must retrieve it from disk. This is referred to as a read miss. A read miss incurs increased I/O response time due to the innate mechanical delays of hard disk drives.

Since workloads with high Symmetrix cache read-hit rates are already serviced at memory access speed, deploying them on Flash drive technology may show a small benefit. Workloads with low Symmetrix cache read-hit rates that exhibit random I/O patterns, with small I/O requests of up to 16 KB, and require high transaction throughput benefit most from the low latency of Enterprise Flash drives.

Figure 52  Magnetic disk capacity and performance history

**DB2 workloads best suited for Flash drives**
Log and database file placement recommendations

As mentioned earlier, all application I/Os to the Symmetrix subsystem pass through Symmetrix cache. Under normal operational circumstances, writes are always serviced immediately after being received in cache. Read response times, however, can differ considerably depending on whether the data is found in cache (read hit) or needs to be read from disk (read miss). In the case of a read miss, the physical characteristics of the disk being read from and how busy it is affects the response time significantly. It is also common for databases to issue large I/Os (larger than 32 KB) when sequential reads are involved, such as full table scans and backups. When the array identifies sequential read streams, it attempts to use prefetch to bring the requested data into cache ahead of time, increasing the cache read hit for this data. Based on this, it can be concluded that the most obvious advantage for the Flash drives, by virtue of their extremely low latency, is to service small I/O (16 KB and less) random read-miss workloads.

Database log file activity is mostly sequential writes and because writes are serviced from cache, logs are not necessarily good candidates for placement on Flash drives and can be placed on HDD to leave more capacity for other files on the Flash drives.

Database datasets supporting an OLTP workload are good candidates for placement on Flash drives. Typically, an OLTP workload generates small I/O, random read/write activity to the table space datasets. When a random OLTP application creates a lot of read-miss activity, Flash drives can improve the performance of the workload many times over HDD.

Temp files and indices

Database temp files and indices tend to also create random read activity. For example, if multiple queries sort data on disk simultaneously, the temp files workload becomes relatively random and therefore benefits from placement on Flash drives. In a similar way, indices are commonly B-tree structures, and when they don’t fit in cache, they benefit from being positioned on Flash drives.

Other database file types can be considered for Flash drives according to their I/O profile and the requirement for high throughput and low random read response time.
Flash drives and storage tiering

While it may be feasible to have a complete DB2 system deployed on high-cost Flash drives, it may not be economically justifiable if the database is very large. Moreover, parts of the database may be hot (high intensity access) and other parts cold (rarely used). An ideal layout would be to have the hot data on Flash drives and the cold data on spinning media. If the access patterns are consistent, that is to say, certain parts of the database are always hot and certain parts are always cold, this tiered approach can be implemented very easily.

However, it is probable that data access patterns are more volatile, and they tend to change over time. In many cases, the age of the data is a key factor that determines what is hot and what is not. In this case, table partitioning might be able to help.

Table partitioning allows parts of a table to reside in different table spaces based on certain predefined key values. The table space partitions can be deployed on different storage tiers according to the partition access requirements.

The goal with this approach is to partition the table in such a way that the various partitions have known comparative access profiles. This is usually accomplished by partitioning by date range since it is common that the newest data is the most frequently accessed, and the older the data, the less it is retrieved.

Table partitioning is commonly used in data warehouses where it enhances index and data scans. However, with a tiered storage strategy in mind, customers should consider the advantage of using table partitioning for OLTP applications. While partitioning allows the distribution of the data over multiple storage tiers, including Flash drives, it does not address the data movement between tiers. Solutions for data migration between storage tiers are available from some database applications, volume managers, or storage management software. An example of using table partitioning is shown in Figure 53 on page 254.
Data placement considerations

Placement of the data on the physical spindles can potentially have a significant impact on DB2 system performance. Placement factors that affect database performance include:

- Hypervolume selection for specific database files on the physical spindles themselves.
- The spread of database files across the spindles to minimize contention.
- The placement of high I/O devices contiguously on the spindles to minimize head movement (seek time).
- The spread of files across the spindles and back-end directors to reduce component bottlenecks.

Each of these factors is discussed in the sections below.

Disk performance considerations

As shown in Figure 54 on page 255, there are five main considerations for spindle performance:
Actuator Positioning (Seek Time) — This is the time it takes the actuating mechanism to move the heads from their present position to a new position. This delay averages a few milliseconds and depends on the type of drive. For example, a 15 k drive has an average seek time of approximately 3.5 ms for reads and 4 ms for writes. The full disk seek is 7.4 ms for reads and 7.9 ms for writes.

Note: Disk drive characteristics can be found at www.seagate.com.

Rotational latency—This is because of the need for the platter to rotate underneath the head to correctly position the data that must be accessed. Rotational speeds for spindles in the Symmetrix subsystem range from 7,200 rpm to 15,000 rpm. The average rotational latency is the time it takes for one half of a revolution of the disk. In the case of a 15 k drive, this would be about 2 ms.

Interface speed—This is a measure of the transfer rate from the drive into the Symmetrix cache. It is important to ensure that the transfer rate between the drive and cache is greater than the drive’s rate to deliver data. Delay caused by this is typically a very small value, on the order of a fraction of a millisecond.
Areal density—This is a measure of the number of bits of data that fits on a given surface area on the disk. The greater the density, the more data per second can be read from the disk as it passes under the disk head.

Drive cache capacity and algorithms—Newer disk drives have improved read and write algorithms, as well as cache, to improve the transfer of data in and out of the drive and to make parity calculations for RAID 5.

Delay caused by the movement of the disk head across the platter surface is called seek time. The time associated with a data track rotating to the required location under the disk head is referred to as rotational latency or delay. The cache capacity on the drive, disk algorithms, interface speed, and the areal density (or zoned bit recording) combines to produce a disk transfer time. Therefore, the time it takes to complete an I/O (or disk latency) consists of the seek time, the rotational delay, and the transfer time.

Data transfer times are typically on the order of fractions of a millisecond. Therefore, rotational delays and delays because of repositioning the actuator heads are the primary sources of latency on a physical spindle. Additionally, rotational speeds of disk drives have increased from top speeds of 7,200 rpm up to 15,000 rpm, but still average on the order of a few milliseconds. The seek time continues to be the largest source of latency in disk assemblies when using the entire disk.

Transfer delays are lengthened in the inner parts of the drive—more data can be read per second on the outer parts of the disk surface than by data located on the inner regions. Therefore, performance is significantly improved on the outer parts of the disk. In many cases, performance improvements of more than 50 percent can sometimes be realized on the outer cylinders of a physical spindle. This performance differential typically leads customers to place high I/O objects on the outer portions of the drive.

Performance differences across the drives inside the Symmetrix array are significantly smaller than the stand-alone disk characteristics would attest. Enginuity operating environment algorithms, particularly the algorithms that optimize ordering of I/O as the disk heads scan across the disk, greatly reduce differences in hypervolume performance across the drive. Although this smoothing of disk latency may actually increase the delay of a particular I/O, overall performance characteristics of I/Os to hypervolumes across the face of the spindle is better and more uniform.
Hypervolume contention

Disk drives are capable of receiving only a limited number of read or write requests before performance degrades. While disk improvements and cache, both on the physical drives and in disk arrays, have improved disk read and write performance, the physical devices can still become a critical bottleneck in DB2 system environments. Eliminating contention on the physical spindles is a key factor in ensuring maximum DB2 performance on Symmetrix arrays.

Contention can occur on a physical spindle when I/O (read or write) to one or more hypervolumes exceeds the I/O capacity of the disk. While contention on a physical spindle is undesirable, migrating high I/O data onto other devices with lower utilization can rectify this type of contention. This can be accomplished using a number of methods, depending on the type of contention that is found. For example, when two or more hypervolumes on the same physical spindle have excessive I/O, contention may be eliminated by migrating one of the hypervolumes to another, lower utilized physical spindle. One method of reducing hypervolume contention is careful layout of the data across the physical spindles on the back end of the Symmetrix subsystem.

Hypervolume contention can be found in a number of ways. DB2-specific data collection and analysis tools such as the DB2 Performance Monitor, as well as host tools, can identify areas of reduced I/O performance in the database. Additionally, EMC tools such as Performance Manager can help to identify performance bottlenecks in the Symmetrix array. Establishing baselines of the system and proactive monitoring are essential to maintain an efficient, high-performance database.

Commonly, tuning database performance on the Symmetrix system is performed post-implementation. This is unfortunate because with a small amount of up-front effort and detailed planning, significant I/O contention issues could be minimized or eliminated in a new implementation. While detailed I/O patterns of a database environment are not always well known, particularly in the case of a new system implementation, careful layout consideration of a database on the back end of the Symmetrix system can save time and future effort in trying to identify and eliminate I/O contention on the disk drives.
Maximizing data spread across the back end

Data on the Symmetrix array should be spread across the back-end directors and physical spindles before locating data on the same physical drives. By spreading the I/O across the back end of the array, I/O bottlenecks in any one array component can be minimized or eliminated.

Considering recent improvements in the Symmetrix component technologies such as CPU performance on the directors and the Virtual Matrix Architecture, the most common bottleneck in new implementations is with contention on the physical spindles and the back-end directors. To reduce these contention issues, a detailed examination of the I/O requirements for each application that will utilize the Symmetrix storage should be made. From this analysis, a detailed layout that balances the anticipated I/O requirements across both back-end directors and physical spindles should be made.

Before data is laid out on the back-end of the Symmetrix subsystem, it is helpful to understand the I/O requirements for each of the file systems or volumes that are being laid out. Many methods for optimizing layout on the back-end directors and spindles are available. One time-consuming method involves creating a map of the hypervolumes on physical storage, including hypervolume presentation by director and physical spindle, based on information available in EMC ControlCenter®. This involves documenting the environment using a tool such as Excel, with each hypervolume marked on its physical spindle and disk director. Using this map of the back end and volume information for the database elements, preferably categorized by I/O requirement (high/medium/low, or by anticipated reads and writes), the physical data elements and I/Os can be evenly spread across the directors and physical spindles.

This type of layout can be extremely complex and time-consuming. Additional complexity is added when RAID 5 hypervolumes are added to the configuration. Since each hypervolume is really placed on either four or eight physical spindles in RAID 5 environments, trying to uniquely map out each data set or database element is beyond what most customers feel is valuable. In these cases, one alternative is to rank each of the database elements or volumes in terms of anticipated I/O. Once ranked, each element may be assigned a hypervolume in order on the back end. Since BIN file creation tools almost always spread contiguous hypervolume numbers across different elements of the back end, this method of assigning the
ranked database elements usually provides a reasonable spread of I/O across the spindles and back-end directors in the Symmetrix controller. Combined with Symmetrix Optimizer, this method of spreading the I/O is normally effective in maximizing the spread of I/O across Symmetrix components.

**Minimizing disk head movement**

Perhaps the key performance consideration controllable by a customer when laying out a database on the Symmetrix array is minimizing head movement on the physical spindles. Head movement is minimized by positioning high I/O hypervolumes contiguously on the physical spindles. Disk latency that is caused by interface or rotational speeds cannot be controlled by layout considerations. The only disk-drive performance considerations that can be controlled are the placement of data onto specific, higher performing areas of the drive (discussed in a previous section), and the reduction of actuator movement by trying to place high I/O objects in adjacent hypervolumes on the physical spindles.

One method, described in the previous section, describes how volumes can be ranked by anticipated I/O requirements. Utilizing a documented map of the back-end spindles, high I/O objects can be placed on the physical spindles, grouping the highest I/O objects together. Recommendations differ as to whether placing the highest I/O objects together on the outer parts of the spindle (that is, the highest performing parts of a physical spindle) or in the center of a spindle are optimal. Since there is no consensus to this, the historical recommendation of putting high I/O objects together on the outer part of the spindle is still a reasonable suggestion. Placing these high I/O objects together on the outer parts of the spindle should help to reduce disk actuator movement when doing reads and writes to each hypervolume on the spindle, thereby improving a controllable parameter in any data layout exercise.

**Other layout considerations**

Besides the layout considerations described in previous sections, a few additional factors may be important to DBAs or storage administrators who want to optimize database performance. Some additional configuration factors to consider include:

- Implementing DB2 with SRDF/S
Creating DB2 copies using TimeFinder/Mirror

Creating DB2 copies using TimeFinder/Clone or TimeFinder/Snap

These additional layout considerations are discussed in the next sections.

Implementing DB2 with SRDF/S

When DB2 for z/OS is implemented within an SRDF configuration, it is recommended to stripe the active logs using VSAM striping. The active logs are the key delaying factor during a transaction as the propagation delay affects transactions directly because writes to the logs are synchronous with respect to the transaction when a commit is executed. In contrast, data writes from the buffer pool are mostly asynchronous to the transaction execution, and any delay introduced to these writes does not affect transaction performance.

Striping the logs as described in the section “VSAM striping for DB2 active logs” on page 262 is beneficial since it builds write queues for SRDF on more than one hypervolume, and these writes can be dispatched (in general) in parallel.

TimeFinder and sharing spindles

DB2 system cloning is useful when DBAs wish to create backups or images of a system for other purposes. A common question when laying out a DB2 system is whether the Symmetrix target volumes for TimeFinder/Mirror or TimeFinder/Clone should share the same physical spindles as the production volumes, or whether they should be isolated on separate physical disks. There are pros and cons to either of the two solutions. The optimal solution generally depends on the anticipated workload.

The primary benefit of spreading the target volumes across all physical spindles is performance. By spreading I/Os across more spindles, there is a reduced chance of developing bottlenecks on the physical disks. Workloads that utilize the target volumes, such as backups and reporting databases, may generate high I/O rates. Spreading this workload across more physical spindles may significantly improve performance in these environments.
The main drawbacks to spreading target volumes across all spindles in the Symmetrix array are that the resynchronization process may cause spindle contention and that workloads on the target volumes may negatively impact production database performance. When re-synchronizing the target volumes, data is read from the production hypervolumes and copied into cache. From there, it is destaged to the targets. When the physical disks contain both production volumes and TimeFinder targets, the synchronization rates can be greatly extended because of increased seek times due to the head reading one part of the disk and writing to another. The other drawback to sharing physical disks is the increased workload on the spindles that make impact performance on the production volumes. Sharing the spindles increases the chance that contention may arise, decreasing DB2 performance.

Determining the appropriate location for the TimeFinder target volumes, sharing the same physical spindles or isolated on their own disks, depends on customer preference and workload. In general, it is recommended that the TimeFinder target volumes share the same physical spindles. However, in cases where the target volume synchronization and utilization negatively impact applications (such as, DB2 systems that run 24/7 with high I/O requirements), it may be beneficial for the TimeFinder target volumes to be isolated on their own physical disks.

**TimeFinder/Snap and TimeFinder/Clone**

TimeFinder/Snap provides many of the benefits of full volume replication techniques, such as TimeFinder/Mirror or TimeFinder/Clone, but at greatly reduced costs. However, two performance considerations must be taken into account when using TimeFinder/Snap to make database clones for backups or other business continuity functions. The first of these impacts, Copy On First Write (COFW), derives from the need for data to be copied from the production hypervolumes to the SAVE area as data is changed. This penalty affects only writes to the source volumes. With Enginuity 5772 and later, the COFW penalty is mostly eliminated since the copying of the track is asynchronous to the write activity. The second potential penalty happens when snaps are accessed. Tracks that have not been copied to the save area have to be read from the source volumes. This additional load on the source volumes can have an impact on heavily loaded systems.
VSAM striping for DB2 active logs

Sequential dataset striping was introduced with DFSMS 1.1. Implementation of this demonstrated significant throughput for large sequential accesses. VSAM striping introduced with DFSMS R10. When used for DB2 active logs in the EMC lab, reductions in runtimes were realized. This shortened batch windows in EMC testing.

When using VSAM striping, there is a maximum of 16 stripes per VSAM dataset and 255 extents per stripe. To enable VSAM striping, define the active log datasets with STRIPE COUNT > 1. RESET/REUSE is not supported.

The following are required for defining VSAM striping. These are the SMS class definitions:

- Dataset Name Type = EXTENDED
- Sustained Data Rate (MB/sec) > 0 (for example, 40)

The define cluster should include:

- Volumes=(** *) (4 stripes)

Figure 55 on page 262 shows an example of DB2 active log being striped across 4 devices.

DB2 V9 and DFSORT

The DB2 V9 utilities use DFSORT exclusively. Even if DFSORT is not the default sort in the z/OS system, it is required for the following DB2 V9 utilities:

- LOAD
Remote replication performance considerations

This section concerns the performance considerations when a DB2 system is replicated from one location to another remote location such as described in Chapter 7. The mitigation of the impact of synchronous replication on the database and the reduction in bandwidth requirements for the solution are reviewed.

Synchronous replication considerations

Symmetrix synchronous replication involves the simultaneous duplication of changes made at the source database to another database on a different Symmetrix system, usually in a different location. Writes to the database datasets and log files are not completed until the changed data has made it to the remote array and an acknowledgement returned to the local array. This process delays all writes to the database and logs and significantly impacts performance, particularly when longer distances are involved.
Buffer pool I/O

The writes to the data volumes are mainly asynchronous to the application threads and are usually triggered by buffer pool thresholds being hit and, for the most part, are minimally affected by the synchronous replication. The actual impact is seen as a marginal slowing down of the buffer pool I/O cleaning process.

 Writes from the buffer pool to the datasets because of dirty page steals are synchronous to the application thread, and service times for applications affected by the steals will be extended. The best way to avoid buffer pool steals is to adjust the buffer pool thresholds (VDWQT(0,40) is generally a good starting point).

Active log I/O

 Writes to the active log files are synchronous with respect to the application thread. The log writes are usually initiated by a COMMIT or ROLLBACK SQL statement. After the COMMIT or ROLLBACK is issued, the application waits for a positive acknowledgement that the write to the log is complete before continuing. Any delay introduced to the log write process because of remote replication delays the transaction. While this delay may be small (5 to 10 ms, for example) in a heavy write environment it can affect overall transaction throughput.

 Unfortunately there is little that can be done to mitigate the latency that is introduced by synchronous replication. The three things that can be done:

- Use VSAM striping to spread the log I/O across more Symmetrix hypervolumes as described in “VSAM striping for DB2 active logs” on page 262. This allows more parallel log operations to take place on the link.
- Use PAVs with the active log volumes to reduce IOSQ time for the log access.
- Use Enginuity 5772 or later for distances between 50km and 100 km and enable the SiRT technology (Single round trip journaling) and which can reduce the latency associated log writes at these distances.
Bandwidth reduction

The bandwidth required for a given remote replication solution is a key component in the design and a continuing, repeated cost. Any reduction in the amount of data transmitted without compromising the end solution is an important goal for the design. This section discusses ways that can reduce the quantity of data transmitted. While bandwidth reduction is not a key factor in synchronous solution design, it is a critical factor in an asynchronous solution design, especially over long distances.

Temp space replication

DB2 system in DSNDB07 temp space is used for temporary data, sorts, and intermediate results of complex SQL statements. The changes made to temp space are therefore not required at the remote location. To prevent replication of temp changes to the remote site, isolate temporary table spaces to volumes that are not being replicated. These volumes should be dedicated to temp space and not contain any information that is required at the remote site.

DB2 use of PAVs on Symmetrix arrays

Parallel access volumes, or PAVs allow processing of given volume through multiple UCBs on the host. This increases the level of parallelism that the host can direct at a given volume. PAVs can measurably reduce IOSQ (I/O supervisor queue) time.

PAVs come in three flavors, dynamic, static and hyper. While this is not the document to dive deep into all three of these, it is enough to know that all three of them can dramatically reduce IOSQ time in the right situation.

Where PAVs really help is in a cache-friendly workload. I/Os that are satisfied by cache access are executed without having to go to the disk and thus can be parallelized in the Symmetrix controller. For instance, all writes are written directly to cache and do not immediately require disk access. Thus, write workload is ideal for access using PAVs.
PAVs are also beneficial when using Flash drives. EFD services times for reads and writes are so low it is beneficial to direct more I/Os in parallel to the Flash Drives and thus avoid queuing in the host. In addition, unlike rotating disks, Enterprise Flash drives can perform multiple operations in parallel.

Similarly, when a volume presented to z/OS has more than one spindle servicing I/O (such as in a RAID 5, RAID 6, or RAID 10 device), I/O requests have the potential to be executed in parallel in the Symmetrix array on the different spindles underpinning that particular RAID protection. These kind of RAID configurations are ideally suited for PAV access.

It should be noted that, if there is only a single spindle servicing I/O on the storage controller and if it is a cache-hostile workload (random read miss), the effect of PAVs is to move the queuing to the controller from the host, manifesting in increased DISC (disconnect) time.

### Extended address volumes

Enginuity 5874 introduced the ability to create and utilize hypervolumes that can be up to 262,668 cylinders in size. These large volumes are supported as IBM 3390 format with a capacity of up to 223 GB and are configured as 3390 Model As. This large volume size matches the capacity announced in z/OS 1.10 for 3390 Extended Address Volumes (EAV). IBM defines an EAV to be any volume that is greater than 65,520 cylinders, hence any subset volume size greater than 65,520 cylinders and less than or equal to 262,668 cylinders is an EAV. An EAV is currently limited to 262,668 cylinders but has a defined architectural limit of 268,434,453 cylinders.

With Enginuity 5874, large volumes may be configured and used in a similar manner to the old, regular devices that users are already familiar with. While large volumes can co-exist alongside the older volumes, there are limitations to their use imposed by certain access methods (operating system restrictions) and other independent vendor’s software.

The collective reference to the space above 65,520 cylinders is known as the extended addressing space (EAS), while the space below 65,520 cylinders is referred to as the base addressing space. Today, the major exploiter of the EAS portions of EAVs are applications using any form of VSAM (KSDS, RRDS, ESDS, and linear). This covers DB2, IMS, CICS, zFS, and NFS. A few restrictions are notable with z/OS 1.10.
with respect to VSAM datasets not supported in the EAS. They are: catalogs, paging spaces, VVDS datasets, and those with KEYRANGE or IMBED attributes.

Large volumes offer the ability to consolidate many smaller volumes onto a single device address. This, in turn, goes a long way in solving the sprawling device configurations which placed users at the Logical Partition (LPAR) device limit of 65,280 devices. In fact, the two main culprits of device sprawl in many environments are: The excessive and prolonged use of very small volume sizes (3390-3, 3390-9 etc.), and the significant and growing business continuity requirements as business applications proliferate.

Not only are large volumes an important component in the device consolidation strategy, but it is quickly recognized as an even more vital component in providing the very high z/OS system capacities required by the newer class of applications running on the ever more powerful processors. In short, even with a few addresses configured with these large capacities, the result is very large system capacities.

One of the stated goals of large volumes and the accompanying reduction of device addresses required to support them is the overall simplification of the storage system, and hence, the reduction of storage management costs. This has a direct impact on storage personnel productivity and proves to be a strong determinant in its adoption.

Last, but certainly not least, is the reduction of processor resources associated with multi-volume usage and processing since datasets and clusters can now be wholly contained within large enough single extents, thereby relegating multi-volume extents and their drawbacks to something of user preference and choice rather than necessity. The reduction in frequency of OPEN/CLOSE/End of Volume processing helps expedite many currently long-running batch jobs.

At this time, it is not recommended to put high-demand production DB2 systems on very large EAV volumes since there may be insufficient spindles to service the I/O workload. EAV volumes may be applicable for DB2 systems in test or QA where performance criteria may not be so important as in production applications.
This chapter presents these topics:

- **Overview** ........................................................................................................ 270
- **DB2 use in an SAP environment** .................................................................. 271
Overview

The z/OS processing environment provides a highly scalable and highly available processing platform. Parallel sysplex implementations provide high scalability and availability using DB2 data sharing capabilities. The combination of Parallel sysplex and DB2 data sharing functions make it a preferred choice for providing database server functions for large SAP implementations.

SAP uses a multi-tier implementation approach, where client processes interact with application servers. Application servers in turn, interact with the database server. All data is stored on the database server. Figure 56 on page 270 shows an SAP-processing environment using DB2 for z/OS as the database server.

**Figure 56 High-level SAP processing environment**

Characteristics of an SAP environment that set it apart from a typical DB2 application are:

- Large number of DB2 objects—When combined with the DB2 catalog objects, the total number of objects to be managed within the DB2 system can be over 70,000.
**DB2 use in an SAP environment**

SAP on z/OS typically uses DB2 as the database server. DB2 can manage large amounts of data, delivering high performance and high levels of availability, integrity, and security. These features of DB2 provide an environment that can handle SAP’s complex application design and the large number of database objects. It is recommended that SAP run on a dedicated DB2 system. The dedicated DB2 system is a fundamental requirement to providing storage processor support for managing large SAP environments using DB2 for z/OS.

**Storage management**

Standard DB2 backup, recovery, and cloning methods are difficult to manage and time consuming in an SAP environment. This is due to the large number of DB2 objects that SAP introduces into the DB2 system which need to be backed up and recovered together. EMC provides alternative solutions to traditional DB2 utilities for cloning SAP systems, providing backup and recovery and providing disaster recovery or business continuance.

**Cloning SAP environments**

EMC technology enables creation of an instantaneous point-in-time copy of an SAP system. The cloned copy is an identical environment to its source, and can be used for other processing purposes, such as backup, recovery, offline reporting, and testing.
Backup and recovery

Backup and recovery operations using DB2 utilities require intervention by experienced personnel and can be labor intensive in a large DB2 system. Recovery of SAP systems is especially complex because the entire SAP system is a large referential set. All data in the set needs to be recovered together and to the same recovery point.

Dynamic manipulation of objects and application-maintained referential integrity further complicates recovery efforts. Traditional DB2 recovery techniques require multiple passes of the data, which can greatly impact recovery times. Such techniques and are generally unworkable in an SAP environment due to the time required to recover all objects. EMC hardware and software can be used to make the process faster and more effective.

Holistic business continuity for SAP

Application integration and data integration processes drives a need for a holistic business continuity approach. Storage systems are the common denominator to provide holistic solutions. EMC Symmetrix examples include:

- TimeFinder/Mirror and TimeFinder/Clone
- Consistency Technology
- SRDF/S
- SRDF/AR
- SRDF/A
- SRDF/Star
- SRDF/EDP

Foundation for holistic business continuity solutions

- Local Replication – EMC TimeFinder Family—This is the ability to replicate data within a storage processor. The replication is performed with volume mirrors or copy processes. Full-volume or virtual copies can be made. Consideration must be given to maintain dependent-write consistency.
- Remote Replication—SRDF—This is the ability to replicate data over a distance. Synchronous replication and asynchronous replication can be used. Data relationship domains are...
maintained using embedded consistency technology, write ordering, or I/O time stamping. Storage system replication is the foundation for disaster restart. Consistency technology is required to preserve data integrity.

For more information on EMC solutions for SAP environments on z/OS, refer to the other chapters in this TechBook.
This appendix contains information about related documents.
Related documents

The information in this document was compiled from the following sources:

- DB2 10 for z/OS Administration Guide, SC19-2968-07, IBM Corporation
- DB2 10 for z/OS Data Sharing: Planning and Administration, SC19-2972-05, IBM Corporation
- DB2 10 for z/OS Utility Guide and Reference, SC19-2984-06, IBM Corporation
- DB2 10 for z/OS Installation and Migration Guide, GC19-2974-08, IBM Corporation
- DB2 10 for z/OS Application Programming & SQL Guide, SC19-2969-04, IBM Corporation
- EMC TimeFinder/Mirror for z/OS Product Guide
- EMC TimeFinder/Clone Mainframe SNAP Facility Product Guide
- EMC TimeFinder Utility for z/OS Product Guide
- EMC Symmetrix Remote Data Facility (SRDF) Product Guide
- EMC SRDF Host Component for z/OS Product Guide
- EMC Consistency Group for z/OS Product Guide
- EMC GDDR for SRDF/Star Product Guide
The REXX sample scripts provided in this appendix are to assist you with the cloning steps that are detailed in Chapter 3.
Overview

The REXX sample scripts provided in this appendix are to assist you with the cloning steps that are detailed in Chapter 3.
REXX Example #1

This appendix provides a REXX example to assist with step 1 in the Section “Creating objects on the target DB2” on page 94.

```rexx
/*----------------------------------------REXX----------------------------------------*/
/* */
/* DESCRIPTION : TAKES AN INPUT DDL AND GENERATES AN OUTPUT DDL */
/* WITH THE OBID CLAUSE FOR EACH 'CREATE TABLE' STATEMENT */
/* THAT IS FOUND IN THE 'SSID' system. */
/* */
/* PARMS : SSID - DB2 system ID */
/* */
/* I/O : DDLIN - INPUT DDL */
/* DDLOUT - GENERATED OUTPUT DDL */
/* */
/*-------------------------------------------------------------------*/

PARSE ARG SSID
NUMERIC DIGITS 15
DB2SQL_RC = 0
SQLACPT = '0 100'
SQLID = ''
OUT_CNT = 1 /* OUTPUT DDL LINE COUNTER */

/* READ INPUT DDL */
ADDRESS TSO "EXECIO * DISKR DDLIN (FINIS STEM LINE."

CALL CLEAN_INPUT
CALL DB2_CONNECT SSID
IF (RESULT ¬= 0) THEN EXIT(8)

/* FOR EACH DDL STATEMENT DO: */
DO I = 1 TO SC.0
   CALL GET_NEXT_STMT I
   /* IS THIS A 'SET CURRENT SQLID' STATEMENT ?? */
   PARSE UPPER VAR NEXT_STMT "SET CURRENT SQLID" "=" "" CURR_SQL "" ";"
   IF (CURR_SQL ¬= '') THEN SQLID = CURR_SQL
   /* IS THIS A 'CREATE TABLE' STATEMENT ?? */
   PARSE UPPER VAR NEXT_STMT PREFIX "CREATE TABLE " TB_NAME SUFFIX
   IF (PREFIX ¬= NEXT_STMT) THEN
      DO
         /* 'CREATE TABLE' STATEMENT */
         STMT = NEXT_STMT
         CALL FIND_OBID STMT
         IF (OBID_FOUND = 1) THEN
            DO
               SAY 'WARNING - OBID EXISTS IN INPUT DDL FOR TABLE 'TB_NAME
               CALL COPY_STMT I
            END
         END
      END
   END

END
```
ELSE DO
    CALL DB2_SELECT_OBID TB_NAME
    IF (RESULT = 0) THEN CALL CREATE_NEW_STMT I
    ELSE CALL COPY_STMT I
END
END
ELSE /* NOT A 'CREATE TABLE' STATEMENT */
    CALL COPY_STMT I
END
CALL DB2_DISCONNECT
OUT_LINE.0 = OUT_CNT
ADDRESS TSO "EXECIO * DISKW DDLOUT (STEM OUT_LINE. FINIS"
EXIT 0

/*-----------------------------------------------*/
/* CREATE A NEW STEM VAR WITH THE DDL STATEMENT WITH : */
/* - NO COMMENTS */
/* - SEPARATE STATEMENTS IN SEPARATE LINES */
/* - SEMI-COLONS IN SEPARATE LINES */
/*-----------------------------------------------*/

CLEAN_INPUT :
J = 1 /* OUTPUT DDL LINE COUNTER */
S = 1 /* ';' VECTOR LINE COUNTER */
DO I=1 TO LINE.0
    TEMP = SPACE(LINE.I)
    /* SCRATCH THE COMMENT PART OF THE LINE */
    PARSE VAR TEMP CURR_LINE"--"COMMENT
    /* IF LINE CONTAINS ';' ONLY */
    IF (SPACE(CURR_LINE) = ';') THEN
        DO
            CLEAN_LINE.J = CURR_LINE
            SC.S = J
            S = S + 1
            J = J + 1
        END
    IF (CURR_LINE ¬= '') THEN
        DO
            SUFFIX='SUFFIX'
            DO UNTIL (SUFFIX='')
                PARSE VAR CURR_LINE PREFIX';'SUFFIX
                IF (PREFIX = CURR_LINE) THEN
                    DO
                        /* NO ';' IN LINE */
                        CLEAN_LINE.J = CURR_LINE
                        J = J + 1
                    END
                ELSE IF (PREFIX ¬= '') THEN
                    DO
                        /* ';' FOUND IN LINE */
                        CLEAN_LINE.J = SPACE(PREFIX)
                        J = J + 1
                        CLEAN_LINE.J = ";"
                    END
                END
            END
        END
    END
DO
    SUFFIX='SUFFIX'
    DO UNTIL (SUFFIX='')
        PARSE VAR CURR_LINE PREFIX';'SUFFIX
        IF (PREFIX = CURR_LINE) THEN
            DO
                /* NO ';' IN LINE */
                CLEAN_LINE.J = CURR_LINE
                J = J + 1
            END
        ELSE IF (PREFIX ¬= '') THEN
            DO
                /* ';' FOUND IN LINE */
                CLEAN_LINE.J = SPACE(PREFIX)
                J = J + 1
                CLEAN_LINE.J = ";"
            END
        END
    END
END
SC.S = J
S = S + 1
J = J + 1
END
CURR_LINE = SUFFIX
END
END
END
CLEAN_LINE.0 = J
SC.0 = S - 1
RETURN

 /*-----------------------------------------------------------------*/
/* GENERATE THE NEXT DDL STATEMENT FROM THE STEM VAR CLEAN_LINE */
 /*-----------------------------------------------------------------*/
GET_NEXT_STMT :
PARSE ARG I
PRE_I= I - 1
IF (I = 1) THEN STMT_STRRT = 1 ELSE STMT_STRRT = SC.PRE_I + 1
STMT_END = SC.I
NEXT_STMT = ""
DO J = STMT_STRRT TO STMT_END
 _ NEXT_STMT = NEXT_STMT || CLEAN_LINE.J || ' '
END
RETURN

 /*-----------------------------------------------------------------*/
/* FIND IF CREATE TABLE STATEMENT ALREADY CONTAINS OBID CLAUSE */
 /*-----------------------------------------------------------------*/
FIND_OBID :
PARSE ARG STMT
OBID_FOUND = 0
SUFFIX = 'SUFFIX'
DO UNTIL SUFFIX = ''
PARSE UPPER VAR STMT PREFIX "OBID " OBID_STR SUFFIX
IF (PREFIX = STMT) THEN
DO
  B = DATATYPE(OBID_STR,'NUM')
  IF (B = 1) THEN OBID_FOUND = 1
END
STMT = SUFFIX
END
STMT = SUFFIX
RETURN

 /*-----------------------------------------------------------------*/
/* GENERATE THE OUTPUT 'CREATE TABLE' STATEMENT WITH THE OBID */
 /*-----------------------------------------------------------------*/
CREATE_NEW_STMT :
PARSE ARG I
PRE_I = I - 1
IF (I = 1) THEN STMT_STRRT = 1
ELSE STMT_STRT = SC.PRE_I + 1
STMT_END = SC.I
DO P = STMT_STRT TO STMT_END - 1
  OUT_LINE.OUT_CNT = CLEAN_LINE.P
  OUT_CNT = OUT_CNT + 1
END
OUT_LINE.OUT_CNT = 'OBID ' || OBID
OUT_CNT = OUT_CNT + 1
OUT_LINE.OUT_CNT = ';;'
OUT_CNT = OUT_CNT + 1
RETURN

/**---------------------------------------------------------------*/
/* COPY A NON-CREATE-TABLE FROM INPUT VAR TO OUTPUT VAR           */
/*---------------------------------------------------------------*/
COPY_STMT:
PARSE ARG I
PRE_I = I - 1
IF (I = 1) THEN STMT_STRT = 1
ELSE STMT_STRT = SC.PRE_I + 1
STMT_END = SC.I
DO P = STMT_STRT TO STMT_END
  OUT_LINE.OUT_CNT = CLEAN_LINE.P
  OUT_CNT = OUT_CNT + 1
END
RETURN

/**---------------------------------------------------------------*/
/* CONNECT                                                        */
/*---------------------------------------------------------------*/
DB2_CONNECT:
PARSE ARG SSID
S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
IF (S_RC ≠ 0) THEN
  DO
    SAY 'ERROR - DB2 INTERFACE CONNECTION FAILED'
    RETURN(8)
  END
ADDRESS DSNREXX "CONNECT " SSID
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
RETURN(0)

/**---------------------------------------------------------------*/
/* DISCONNECT                                                     */
/*---------------------------------------------------------------*/
DB2_DISCONNECT:
ADDRESS DSNREXX "DISCONNECT"
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
S_RC = RXSUBCOM('DELETE','DSNREXX','DSNREXX')
RETURN

/**---------------------------------------------------------------*/
/* SELECT */
/*-------------------------------------------------------------------*/
DB2_SELECT_OBID:
PARSE ARG TB_NAME
STMT_SQLID = ''
IF (SQLID ^= '') THEN STMT_SQLID = SQLID
PARSE UPPER VAR TB_NAME CREATOR "." TB
IF (TB ^= '') THEN
   TB_NAME = TB
   IF (CREATOR ^= TB_NAME) THEN STMT_SQLID = CREATOR
END
IF (STMT_SQLID = '') THEN
   DO
      SQL = "SELECT COUNT(*) FROM SYSIBM.SYSTABLES WHERE "
      SQL = SQL || "NAME=''||TB_NAME||''"
      CALL EXEC_SQL SQL
      IF (SQL_VAL = 0) THEN
         DO
            SAY 'ERROR - TABLE 'TB_NAME' DOES NOT EXIST ON 'SSID
            RETURN(8)
         END
      END
      IF (SQL_VAL > 1) THEN
         DO
            SAY 'ERROR - MORE THAN ONE CREATOR FOR TABLE 'TB_NAME
            RETURN(8)
         END
      ELSE DO
         SQL = "SELECT OBID FROM SYSIBM.SYSTABLES WHERE NAME=''||TB_NAME||''"
         CALL EXEC_SQL SQL
         OBID = SQL_VAL
      END
   END
ELSE DO
   SQL = "SELECT OBID FROM SYSIBM.SYSTABLES WHERE NAME=''||TB_NAME||'',
         "||" AND CREATOR=''||STMT_SQLID||''"
   CALL EXEC_SQL SQL
   OBID = SQL_VAL
END
RETURN(0)
EXEC_SQL:
PARSE ARG SQL
/*-------------------------------------------------------------------*/
/* DECLARE CURSOR AND PREPARE STATEMENT */
/*-------------------------------------------------------------------*/
SQLDA. = ''
ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
ADDRESS DSNREXX "EXECSQL PREPARE S1 INTO :SQLDA FROM :SQL"
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
/*-------------------------------------------------------------------*/
/* FETCH THE FIRST */
/*-----------------------------------------*/
ADDRESS DSNREXX "EXECSQL OPEN C1"
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
ADDRESS DSNREXX "EXECSQL FETCH C1 USING DESCRIPTOR :SQLDA"
IF (SQLCODE = 100) THEN RETURN
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
SQL_VAL = SQLDA.1.SQLDATA
IF (SQLCODE != 0) THEN DB2SQL_RC = 4
ADDRESS DSNREXX "EXECSQL CLOSE C1"
IF (WORDPOS(SQLCODE,SQLACPT) = 0) THEN CALL SQLERROR
RETURN

/*-----------------------------------------*/
/* SQL ERROR */
/*-----------------------------------------*/
SQLERROR :
ADDRESS TSO "NEWSTACK"
CALL QUEUE_MESSAGES
Q = QUEUED()
CALL PRINT_MESSAGES
ADDRESS TSO "DELSTACK"
IF (SQLCODE > 0) THEN DB2SQL_RC = 1
ELSE DB2SQL_RC = 8
EXIT DB2SQL_RC
RETURN

/*-----------------------------------------*/
/* PRINT MESSAGES BATCH */
/*-----------------------------------------*/
PRINT_MESSAGES :
DO I = 1 TO Q
   PULL LINE
   SAY LINE
END
RETURN

/*-----------------------------------------*/
/* BUILD SQLCA */
/*-----------------------------------------*/
BUILD_SQLCA :
SQLCAID = 'SQLCA'
SQLCABC = 136
SQLCA = LEFT(SQLCAID,8) || X2C(D2X(SQLCABC,8)) || X2C(D2X(SQLCODE,8)) || X2C(D2X(LENGTH(SQLERRMC),4)) || LEFT(SQLERRMC,70) || LEFT(SQLERRP,8) || X2C(D2X(SQLERRD.1,8)) || X2C(D2X(SQLERRD.2,8)) || X2C(D2X(SQLERRD.3,8)) || X2C(D2X(SQLERRD.4,8)) || X2C(D2X(SQLERRD.5,8)) || X2C(D2X(SQLERRD.6,8)) || LEFT(SQLWARN.0,1) || LEFT(SQLWARN.1,1) || LEFT(SQLWARN.2,1) || LEFT(SQLWARN.3,1) || LEFT(SQLWARN.4,1) || LEFT(SQLWARN.5,1) || LEFT(SQLWARN.6,1) || LEFT(SQLWARN.7,1) || LEFT(SQLWARN.8,1) ||,
LEFT(SQLWARN.9,1) || LEFT(SQLWARN.A,1) || LEFT(SQLSTATE,5)

RETURN

/*-------------------------------------------------------------------*/
/* FORMAT ERROR MESSAGE */
/*-------------------------------------------------------------------*/

QUEUE_MESSAGES :
CALL BUILD_SQLCA
MSG_LLEN = 79
MSG_ROWS = 10
MSG_AREA = X2C(D2X(MSG_LLEN*MSG_ROWS,4))||COPIES(' ',MSG_LLEN*MSG_ROWS)
MSG_LREC = X2C(D2X(MSG_LLEN,8))
ADDRESS LINKPGM "DSNTIAR SQLCA MSG_AREA MSG_LREC"
MSG_LINE_START = 4
DO MESSAGEI = 1 TO MSG_ROWS
  MSG_LINE = SUBSTR(MSG_AREA,MSG_LINE_START,MSG_LLEN)
  MSG_LINE_START = MSG_LINE_START + MSG_LLEN
  IF (MSG_LINE <> '') THEN QUEUE MSG_LINE
END
RETURN
REXX Example #2

This is a REXX example that takes the output of DSNJU004, parses it, and creates a DSNJU003 job to update the new BSDS with the new VCAT name and the new log datasets. This REXX example assists with step 1 in “Creating objects on the target DB2” on page 94 and with the process presented in “Updating the BSDS” on page 116.

```rexx
ADDRESS TSO "NEWSTACK"
ADDRESS ISREDIT
   'MACRO (THLQ) PROCESS'
PARSE UPPER VAR THLQ THLQ
IF THLQ = '' THEN
   DO
      ZEDSMSG = 'MISSING PARAMETER'
      ZEDLMSG = 'TARGET HIGH-LEVEL QUALIFIER IS MISSING'
      ADDRESS ISPEXEC "SETMSG MSG(ISRZ001)"
      EXIT 12
   END

ADDRESS ISPEXEC "VGET (ZUSER) ASIS"
MSG_STATUS = MSG("OFF")
ADDRESS TSO "FREE F(DSNJU004)"
ADDRESS TSO "DELETE 'ZUSER'.TEMP.JU004"'
MSG_STATUS = MSG("ON")
/* ----------------------------------- */
/* CREATE JOB CARD */
/* ----------------------------------- */
ADDRESS TSO "NEWSTACK"
LINE= '//DSNTLOG EXEC PGM=DSNJU003'
QUEUE LINE
LINE= '//SYSUT1 DD DISP=OLD,DSN='THLQ'.BSDS01'
QUEUE LINE
LINE= '//SYSUT2 DD DISP=OLD,DSN='THLQ'.BSDS02'
QUEUE LINE
```
LINE= '//SYSPRINT DD SYSOUT=*'
QUEUE LINE
LINE= '//SYSIN DD *'
QUEUE LINE
LINE=' NEWCAT VSAMCAT='THLQ
QUEUE LINE

ADDRESS ISREDIT
  'X ALL'
  'BNDS 1 122'
  'FIND LOGCOPY1 ALL'
  '(LOG1DS) = FIND_COUNTS'
  'FIND LOGCOPY2 ALL'
  '(LOG2DS) = FIND_COUNTS'
  'DEL X ALL'
  'CHA P"=" 1 " " ALL'

IF (LOG1DS ^= LOG2DS) THEN
  DO
    SAY 'SEVERE ERROR: LOGCOPY1 DS NOT EQUAL LOGCOPY2 DS'
    ADDRESS TSO "DELSTACK"
    EXIT 12
  END

DO I=1 TO LOG1DS BY 1
  IF I<10 THEN
    TEXT = 'LOGCOPY1.DS0' || I
  ELSE
    TEXT = 'LOGCOPY1.DS' || I
  'FIND &TEXT FIRST'
  'ISREDIT (CURROW,CURCOL) = CURSOR'
  'ISREDIT (LOGLINE) = LINE 'CURROW
  PARSE UPPER VAR LOGLINE STARBA ENDRBA DUM1 'DSN=' SHLQ '.LOGCOPY' DUM2
  LINE= ' DELETE DSNAME='SHLQ'.TEXT'
  QUEUE LINE
  LINE= ' NEWLOG DSNAME='THLQ'.TEXT',COPY1,STARTRBA='STARBA','
  QUEUE LINE
  LINE= '
  ENDRBA='ENDRBA
  QUEUE LINE
END

IF LOG2DS > 0 THEN
DO I=1 TO LOG2DS BY 1
  IF I<10 THEN
    TEXT = 'LOGCOPY2.DS0' || I
  ELSE
    TEXT = 'LOGCOPY2.DS' || I
  'FIND &TEXT FIRST'
  'ISREDIT (CURROW,CURCOL) = CURSOR'
  'ISREDIT (LOGLINE) = LINE 'CURROW
  PARSE UPPER VAR LOGLINE STARBA ENDRBA DUM1 'DSN=' SHLQ '.LOGCOPY' DUM2
LINE='DELETE DSNAME='SHLQ'.TEXT'
QUEUE LINE
LINE='NEWLOG DSNAME='THLQ'.TEXT',COPY2,STARTRBA='STARBA','
QUEUE LINE
LINE=''
ENDRBA='ENDRBA'
QUEUE LINE
END
Q=QUEUED()
IF Q>0 THEN
DO
ADDRESS TSO
  "ALLOC F(DSNJU004) DA('"ZUSER".TEMP.JU004') UNIT(SYSDA)
  TRACKS SPACE(1 5) RECFM(F B) LRECL(80)
  DSORG(PS) CATALOG"
ADDRESS TSO
  "EXECIO "Q" DISKW DSNJU004 (FINIS)"
ADDRESS ISPEXEC
  "EDIT DATASET('"ZUSER".TEMP.JU004')"
ADDRESS TSO
  "FREE F(DSNJU004)"
END
ADDRESS TSO "DELSTACK"
EXIT 0
The following REXX could be used to create ALTER statements with commits to potentially long-running ALTER jobs. If commits are not issued, the job might end abnormally.

```rexx
/* REXX */
PARSE ARG SSID NEWSG
SQL="SELECT 'ALTER TABLESPACE '||STRIP(DBNAME)||'.'||STRIP(TSNAME)||" ,
  " PARTITION=0 AND DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')" ,
  " UNION SELECT 'ALTER TABLESPACE '||STRIP(DBNAME)||" ,
  " PARTITION > 0 AND DBNAME NOT IN ('DSNDB07','DSNDB01','DSNDB06')" ,
  " UNION SELECT 'ALTER INDEX '||STRIP(CREATOR)||'.'||STRIP(A.IXNAME)||' PART '||DIGITS(PARTITION)||" ,
  "$ SYSIBM.SYSINDEXPART A, SYSIBM.SYSINDEXES B WHERE "$ ,
  "A.IXNAME=B.NAME AND A.PARTITION = 0 AND DBNAME NOT IN "$ ,
  "('DSNDB07','DSNDB01','DSNDB06') UNION SELECT 'ALTER INDEX '||" ,
  "DIGITS(PARTITION)||" PART '||" ,
  "$ SYSIBM.SYSINDEXPART A, SYSIBM.SYSINDEXES B WHERE "$ ,
  "A.IXNAME=B.NAME AND A.PARTITION > 0 AND DBNAME NOT IN "$ ,
  "('DSNDB07','DSNDB01','DSNDB06')"
NUMERIC DIGITS 15
DB2SQL_RC = 0
COMMIT_RATE=100
SQLACPT = '0 100'
CALL DB2_CONNECT SSID
IF (RESULT <> 0) THEN EXIT(8)
CALL DB2_SELECT
IF (RESULT > 0) THEN DB2SQL_RC = RESULT
CALL DB2_DISCONNECT
EXIT DB2SQL_RC

//CONNECT
/*CONNECT */
/*---------------------------------------*/
DB2_CONNECT :
PARSE ARG SSID
S_RC = RXSUBCOM('ADD','DSNRREXX','DSNRREXX')
ADDRESS DSNREXX "CONNECT " SSID
IF (RC <> 0) THEN
   DO
      SAY 'CONNECTION TO DB2 FAILED, RC='RC' SSID='SSID
      RETURN(8)
   END
```

REXX Example #3

The following REXX could be used to create ALTER statements with commits to potentially long-running ALTER jobs. If commits are not issued, the job might end abnormally.
/* SET ISOLATION LEVEL */
ADDRESS DSNREXX "EXECSQL SET CURRENT PACKAGESET='DSNREXRR'"
RETURN(0)
/*-----------------------------------------------*/
/* DISCONNECT */
/*-----------------------------------------------*/
DB2_DISCONNECT :
ADDRESS DSNREXX "DISCONNECT"
S_RC = RXSUBCOM(‘DELETE’,’DSNREXX’,’DSNREXX’)
RETURN(0)
/*-----------------------------------------------*/
/* SELECT */
/*-----------------------------------------------*/
DB2_SELECT :
/*-------------------------------------------------------------------*/
/* DECLARE CURSOR AND PREPARE STATEMENT */
/*-------------------------------------------------------------------*/
SQLDA. = ''
ADDRESS DSNREXX "EXECSQL DECLARE C1 CURSOR FOR S1"
IF WORDPOS(SQLCODE,SQLACPT) = 0 THEN
DO
   SAY 'DECLARE CURSOR FAILED, SQLCODE='SQLCODE
   RETURN(8)
END
ADDRESS DSNREXX "EXECSQL PREPARE S1 INTO :SQLDA FROM :SQL"
IF WORDPOS(SQLCODE,SQLACPT) = 0 THEN
DO
   SAY 'PREPARE OPERATION FAILED, SQLCODE='SQLCODE
   RETURN(8)
END
/*-------------------------------------------------------------------*/
/* OPEN FETCH LOOP AND CLOSE CURSOR */
/*-------------------------------------------------------------------*/
ROWNUM = 1
ADDRESS DSNREXX "EXECSQL OPEN C1"
IF WORDPOS(SQLCODE,SQLACPT) = 0 THEN
DO
   SAY 'OPEN CURSOR FAILED, SQLCODE='SQLCODE
   RETURN(8)
END
CALL DB2_FETCH
FETCH_RC = RESULT
IF (FETCH_RC > 0) THEN RETURN(FETCH_RC)
DO WHILE (FETCH_RC = 0)
   CALL DB2_FETCH
   FETCH_RC = RESULT
   IF (ROWNUM = COMMIT_RATE) THEN
      DO
         CALL WRITE_DATA ROWNUM
         ROWNUM = 1
      END
   END
END
This glossary contains terms related to disk storage subsystems. Many of these terms are used in this manual.

### A

**actuator**
A set of access arms and their attached read/write heads, which move as an independent component within a head and disk assembly (HDA).

**adapter**
Card that provides the physical interface between the director and disk devices (SCSI adapter), director and parallel channels (Bus & Tag adapter), director and serial channels (Serial adapter).

**adaptive copy**
A mode of SRDF operation that transmits changed tracks asynchronously from the source device to the target device without regard to order or consistency of the data.

**alternate track**
A track designated to contain data in place of a defective primary track. See also “primary track.”

### C

**cache**
Random access electronic storage used to retain frequently used data for faster access by the channel.

**cache slot**
Unit of cache equivalent to one track.
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel director</td>
<td>The component in the Symmetrix system that interfaces between the host channels and data storage. It transfers data between the channel and cache.</td>
</tr>
<tr>
<td>controller ID</td>
<td>Controller identification number of the director the disks are channeled to for EREP usage. There is only one controller ID for a Symmetrix system.</td>
</tr>
<tr>
<td>CKD</td>
<td>Count Key Data, a data recording format employing self-defining record formats in which each record is represented by a count area that identifies the record and specifies its format, an optional key area that may be used to identify the data area contents, and a data area that contains the user data for the record. CKD can also refer to a set of channel commands that are accepted by a device that employs the CKD recording format.</td>
</tr>
<tr>
<td>DASD</td>
<td>Direct access storage device, a device that provides nonvolatile storage of computer data and random access to that data.</td>
</tr>
<tr>
<td>data availability</td>
<td>Access to any and all user data by the application.</td>
</tr>
<tr>
<td>delayed fast write</td>
<td>There is no room in cache for the data presented by the write operation.</td>
</tr>
<tr>
<td>destage</td>
<td>The asynchronous write of new or updated data from cache to disk device.</td>
</tr>
<tr>
<td>device</td>
<td>A uniquely addressable part of the Symmetrix system that consists of a set of access arms, the associated disk surfaces, and the electronic circuitry required to locate, read, and write data. See also “volume.”</td>
</tr>
<tr>
<td>device address</td>
<td>The hexadecimal value that uniquely defines a physical I/O device on a channel path in an MVS environment. See also “unit address.”</td>
</tr>
<tr>
<td>device number</td>
<td>The value that logically identifies a disk device in a string.</td>
</tr>
<tr>
<td>diagnostics</td>
<td>System level tests or firmware designed to inspect, detect, and correct failing components. These tests are comprehensive and self-invoking.</td>
</tr>
<tr>
<td>director</td>
<td>The component in the Symmetrix system that allows transfer of data between the host channels and disk devices. See also “channel director.”</td>
</tr>
</tbody>
</table>
**disk director**  The component in the Symmetrix system that interfaces between cache and the disk devices.

**dual-initiator**  A Symmetrix feature that automatically creates a backup data path to the disk devices serviced directly by a disk director, if that disk director or the disk management hardware for those devices fails.

**dynamic sparing**  A Symmetrix feature that automatically transfers data from a failing disk device to an available spare disk device without affecting data availability. This feature supports all non-mirrored devices in the Symmetrix system.

**ESCON**  Enterprise Systems Connection, a set of IBM and vendor products that connect mainframe computers with each other and with attached storage, locally attached workstations, and other devices using optical fiber technology and dynamically modifiable switches called ESCON Directors. See also “ESCON director.”

**ESCON director**  Device that provides a dynamic switching function and extended link path lengths (with XDF capability) when attaching an ESCON channel to a Symmetrix serial channel interface.

**fast write**  In a Symmetrix subsystem, a write operation at cache speed that does not require immediate transfer of data to disk. The data is written directly to cache and is available for later destaging. This used to be called a DASD fast write.

**FBA**  Fixed Block Architecture, disk device data storage format using fixed-size data blocks.

**frame**  Data packet format in an ESCON environment. See also “ESCON.”

**FRU**  Field replaceable unit: A component that is replaced or added by service personnel as a single entity.
<table>
<thead>
<tr>
<th><strong>G</strong></th>
<th><strong>gatekeeper</strong></th>
<th>A small volume on a Symmetrix storage system used to pass commands from a host to the Symmetrix storage system. Gatekeeper devices are configured on standard Symmetrix disks.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GB</strong></td>
<td><strong>Gigabyte, 10^9 bytes.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>H</strong></td>
<td><strong>head and disk assembly</strong></td>
<td>A field replaceable unit in the Symmetrix system containing the disk and actuator.</td>
</tr>
<tr>
<td></td>
<td><strong>home address</strong></td>
<td>The first field on a CKD track that identifies the track and defines its operational status. The home address is written after the index point on each track.</td>
</tr>
<tr>
<td></td>
<td><strong>hypervolume extension</strong></td>
<td>The ability to define more than one logical volume on a single physical disk device making use of its full formatted capacity. These logical volumes are user-selectable in size. The minimum volume size is one cylinder, and the maximum size depends on the disk device capacity and the emulation mode selected.</td>
</tr>
<tr>
<td><strong>I</strong></td>
<td><strong>ID</strong></td>
<td>Identifier, a sequence of bits or characters that identifies a program, device, controller, or system.</td>
</tr>
<tr>
<td></td>
<td><strong>IML</strong></td>
<td>Initial microcode program load.</td>
</tr>
<tr>
<td></td>
<td><strong>index marker</strong></td>
<td>Indicates the physical beginning and end of a track.</td>
</tr>
<tr>
<td></td>
<td><strong>index point</strong></td>
<td>The reference point on a disk surface that determines the start of a track.</td>
</tr>
<tr>
<td></td>
<td><strong>INLINES</strong></td>
<td>An EMC-internal method of interrogating and configuring a Symmetrix controller using the Symmetrix service processor.</td>
</tr>
<tr>
<td></td>
<td><strong>I/O device</strong></td>
<td>An addressable input/output unit, such as a disk device.</td>
</tr>
<tr>
<td><strong>K</strong></td>
<td><strong>Kilobyte, 1024 bytes.</strong></td>
<td></td>
</tr>
</tbody>
</table>
L

least recently used algorithm (LRU) The algorithm used to identify and make available the cache space by removing the least recently used data.

logical volume A user-defined storage device. In the Model 5200, the user can define a physical disk device as one or two logical volumes.

long miss Requested data is not in cache and is not in the process of being fetched.

longitude redundancy code (LRC) Exclusive OR (XOR) of the accumulated bytes in the data record.

M

MB Megabyte, $10^6$ bytes.

mirrored pair A logical volume with all data recorded twice, once on each of two different physical devices.

mirroring The Symmetrix system maintains two identical copies of a designated volume on separate disks. Each volume automatically updates during a write operation. If one disk device fails, the Symmetrix system automatically uses the other disk device.

P

physical ID Physical identification number of the Symmetrix director for EREP usage. This value automatically increments by one for each director installed in the Symmetrix system. This number must be unique in the mainframe system. It should be an even number. This number is referred to as the SCU_ID.

primary track The original track on which data is stored. See also "alternate track."

promotion The process of moving data from a track on the disk device to cache slot.

R

read hit Data requested by the read operation is in cache.
read miss - Data requested by the read operation is not in cache.

record zero - The first record after the home address.

S

scrubbing - The process of reading, checking the error correction bits, and writing corrected data back to the source.

SCSI adapter - Card in the Symmetrix system that provides the physical interface between the disk director and the disk devices.

short miss - Requested data is not in cache, but is in the process of being fetched.

SSD - Solid State Drive.

SSID - For 3990 storage control emulations, this value identifies the physical components of a logical DASD system. The SSID must be a unique number in the host system. It should be an even number and start on a zero boundary.

stage - The process of writing data from a disk device to cache.

storage control unit - The component in the Symmetrix system that connects the Symmetrix array to the host channels. It performs channel commands and communicates with the disk directors and cache. See also “channel director.”

string - A series of connected disk devices sharing the same disk director.

U

UCB - Unit control block. An in-memory structure containing metadata about an attached device.

unit address - The hexadecimal value that uniquely defines a physical I/O device on a channel path in an MVS environment. See also “device address.”

V

volume - A general term referring to a storage device. In the Symmetrix system, a volume corresponds to single disk device.
W

WLM  Work Load Manager.

write hit  There is room in cache for the data presented by the write operation.

write miss  There is no room in cache for the data presented by the write operation.
Index

Numerics
3390 72

A
ACTIVATE 139, 140
Active logs 22, 23, 77, 87, 132
Adaptive Copy 215, 217
ADRDSSU 80, 88
Archive logs 22, 23, 132
AutoSwap 39, 40, 46, 53, 54, 55

B
BACKUP SYSTEM 155, 156, 157, 158, 165, 177, 178
Bandwidth 196, 200, 201, 215, 237, 244, 265
BC Design Tool 201
BCV 29, 58, 94, 98, 99, 136, 138, 139, 218, 245
Definition 23
Buffer pool 23, 107, 264

C
Cache 238, 239, 240, 241, 247, 251
CAF 25
Cascaded SRDF 41
Catalog Address Space 180
CICS 25
CKD 42, 48
COA 154
Compatible Flash 40, 157, 177
Compatible Native Flash 177
Concurrent Copy 35
Conditional restart 118
Conditional restart record 184
ConGroup 39, 40, 48, 49, 51, 52, 210, 212, 213
Consistency Group 43
Definition 209
Copy pool 155, 160
CSC 37, 38

D
DASD 34
Data compression 201
Data structures 132
Database organization 132
Database recovery 191
Database restart 190, 191
Dataset snap 96
DB2 catalog 22, 23, 24, 52, 93, 94, 95, 98, 104, 105, 111, 112, 113, 117, 118, 119, 120, 122, 124, 126, 132, 133, 136, 144, 166, 169, 171, 185, 235, 270
DB2 catalog indexes 118
DB2 directory 22, 23, 24, 52, 93, 94, 104, 113, 122, 124, 132, 133, 136, 144, 166, 171, 185, 235
DB2 rollback 22
DBD 95
DBID 93, 98, 101, 105, 106
DCM 35
Dependent-write consistency 190, 197, 206, 210, 214, 215
Dependent-write I/O 189, 190, 209
Disaster recovery 189
Index

Disaster restart 189
DISC 266
DMX-4 249
DPR 35
DSN1ADMT 21
DSN1COPY 94
DSN1DBM1 21
DSN1DIST 21
DSN1MSTR 21
DSN1WLM 21
DSNDB01 22, 151, 226
DSNDB06 22, 151, 226
DSNDB07 23, 78, 110, 111, 115, 118, 119, 120, 126, 265
DSNJU003 116, 117, 125
DSNJU004 116
DSNZPARM 78, 111, 112, 113, 123, 124, 128, 150, 171
DSS 237, 248
DSVCI 150
Duplicate Snap 58
Dynamic Cache Partitioning 40, 242
Dynamic Mirror Service Policy 248

E
EAV 35, 74
ECA 48, 49, 50, 60, 137, 138, 139, 140, 147, 178
EMC ControlCenter 258
EMSCSCF 38, 39, 40, 45, 158
Enginuity Consistency Assist 60, 137, 147
ESCON 34, 41, 43
ESTABLISH 136
Extended address volumes 266
EAV 266, 267
EzSM 40, 61

F
FAST 62, 65, 66
FAST VP 68, 85, 86, 87, 88
FBA 42, 48
FCIC 159
FDR Instant® Backup 141
Federated database 197
Fibre Channel 34, 41, 43
FICON 34, 43, 237
Flash drives 249, 250, 251, 252, 253, 266
FlashCopy 35, 37, 155, 156, 157, 158, 161, 165, 176, 177
FlashCopy Image Copy 159
Forward recovery 190

G
GDDR 45, 46, 228
GigE 34, 41, 43
GNS 37, 38
Group Name Service 37, 38

H
HMIGRATE 88
HPGRBLP 178
HPGRBRBA 169
HRECALL 88
HSM 88, 92, 155, 156, 157, 179
HURBA 106

I
IBM 2105 33
IBM 2107 33
IBM 3990 33
ICF catalog 52, 57, 58, 101, 111, 112, 113, 121, 123, 124, 132, 143, 144, 153, 155, 161, 166, 174, 180, 183, 236
IMS 25
IOSLEVEL 49
IOSQ 264, 265
IRLM 123
IRLMPROC 21
ISOBID 93

L
Level ID 92, 93, 98, 104
Locality of reference 200, 203, 217, 218
Log shipping 202, 203
Log suspend 137, 140, 147
LOGONLY 22
M
Mainframe Enabler 72
Migration 79
MSC 39, 40, 45, 216
Multiple Allegiance 33
Multi-Session Consistency 45

N
NOCOPY 160

O
OBID 93, 95
OBIDXLAT 94
OLTP 34, 237, 252
Over-provisioning 70

P
PAV 33, 244, 245, 264, 265, 266
PeopleSoft 26
PERSIST 76
PPRC 35
PREALLOCATE 76
PRECOPY 139
Pre-copy 141
PRIQTY 77, 78
Propagation delay 188, 200
PSID 93, 98, 101, 105
PTAM 198

Q
QSAM 159

R
RACF 57, 58, 59
RAID 266
RAID 1 73, 245, 248
RAID 10 35, 73, 245, 266
RAID 5 35, 73, 246, 248, 256, 258, 266
RAID 6 35, 73, 245, 246, 266
RBR 131, 135, 138, 178
RECOVER LOGONLY 150, 167, 169, 171, 172, 173, 174
Recoverable databases 131
Recovery Expert 161
Recovery point objective 192, 193, 203
Recovery structures 132
Recovery time objective 192, 193
Remote replication 188
REORG 86, 87
ResourcePak Base 36, 37, 38, 39, 42, 43, 216
Restartable databases 131
RESTORE SYSTEM 155, 158, 169, 177, 184, 185
Rolling disaster 203, 207, 209, 211
Rotational latency 255
RPO 192, 193, 194, 202, 203, 215, 218
RRS 25
RTO 189, 192, 193, 194, 199, 202
SRDF Automated Replication, 217
SRDF Consistency Group 42, 46, 47, 48, 49, 54, 190, 206, 209
SRDF consistency group 49
SRDF Host Component 41, 42, 43, 49
SRDF/A 37, 38, 39, 44, 45, 46, 144, 205, 214, 215, 216
SRDF/AR 37, 39, 45, 205, 217, 218
SRDF/Asynchronous 40, 43
SRDF/Automated Replication 40, 45, 59
SRDF/CG 206, 209
SRDF/EDP 41, 223
SRDF/S 144, 205, 259
SRDF/Star 40, 41, 43, 44, 45, 46, 205
Index

SRDF/Synchronous 43
SRDF-ECA 48
SSD 250
STOGROUP 110
Symmetrix Control Facility 76
Symmetrix Management Console 58
Symmetrix Optimizer 243, 248
Symmetrix Priority Control 40
Synchronous replication 200, 205
SYSCOPY 135, 150, 151
SYSLGRNX 150, 151
SYSPITR 177
SYSSTABLES 95
System point-in-time recovery 155

T
Table partitioning 253, 254
TFU 121, 122
Thin device 70, 71, 72, 73, 76, 79
Thin Reclaim Utility 76
Time windows 86
TimeFinder 35, 56, 118, 132, 138, 144, 147, 152, 177, 217
TimeFinder Consistency Groups 144
TimeFinder family of products for z/OS 56
  TimeFinder/CG 40, 57, 59, 60, 140, 144, 147, 190
  TimeFinder/Mirror for z/OS 58
  TimeFinder/Snap for z/OS 57
TimeFinder Utility 57, 58, 100, 104, 114, 121, 141, 143, 149
TimeFinder/CG 138, 139
TimeFinder/Clone 130, 135, 141, 144, 153, 155, 174
TimeFinder/Consistency Groups 130
TSO 25
TSQTY 78

U
Unisphere 58, 72, 85, 86

V
VCAT 110, 118, 120
VDEV 29, 38, 57, 139, 140
Virtual LUN 250
Virtual Provisioning 68, 70, 71, 72, 73, 76
VSAM 23, 77, 119, 142, 264, 266
VSAM striping 264
VTOC 142
VVDS 142

X
XRC 35

Z
zHPF 34