Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

Version 1.3

• Disk-Based Replication
• Online and Offline Database Backups
• Recoverable and Nonrecoverable Database Restores

Roger E. Sanders
Paul Pendle
Dale McInnis
provided by third parties is subject to the terms and conditions of the license that accompanies that software.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of the DB2 programs referenced in this document who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7 CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, the IBM logo, IBM Press, CICS, DB2, developerWorks, MVS, OS/2, RACF, Rational, Redbooks, Tivoli, WebSphere, z/OS and z/VM.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

All other trademarks used herein are the property of their respective owners.
Preface

Chapter 1  Introduction to DB2 for Linux, UNIX, and Windows

The DB2 Family ................................................................. 22
DB2's comprehensive tool set ................................................ 29
  The Control Center ......................................................... 29
  The Command Editor ...................................................... 33
  The Configuration Assistant ............................................. 35
  The Command Line Processor ........................................... 36
Server, instances, and databases .......................................... 39
  The DB2 Administration Server (DAS) instance ................. 40
Objects that make up a DB2 database environment ............... 42
Creating a DB2 LUW database ............................................ 44
  What happens when a DB2 LUW database is created ......... 46
A closer look at table spaces .............................................. 54
  Creating additional table spaces ................................. 58
  Modifying existing table spaces .................................. 59
  Adding new containers to existing automatic storage
    table spaces .............................................................. 61
A closer look at transaction logging .................................... 63
  Logging strategies ....................................................... 66
  Other logging considerations ....................................... 69
Configuring a DB2 LUW database environment .................... 73
  Configuring servers ................................................... 73
  Configuring instances ................................................ 79
  Configuring databases ................................................. 82
# Chapter 2 EMC Foundation Products

Introduction ................................................................................... 88  
Symmetrix hardware and EMC Enginuity features ...................... 90  
Symmetrix VMAX platform ......................................................... 91  
EMC Enginuity operating environment ......................................... 94  
EMC Solutions Enabler base management .................................. 95  
EMC Change Tracker ................................................................. 98  
EMC Symmetrix Remote Data Facility ......................................... 99  
SRDF benefits ........................................................................... 100  
SRDF modes of operation ......................................................... 100  
SRDF device groups and composite groups .............................. 101  
SRDF consistency groups ......................................................... 101  
SRDF terminology .................................................................... 105  
SRDF control operations ......................................................... 107  
Failover and failback operations .............................................. 111  
EMC SRDF/Cluster Enabler solutions ....................................... 114  
SRDF enhancements introduced with Enginuity 5875 .............. 115  
EMC TimeFinder ....................................................................... 117  
TimeFinder/Clone operations ................................................ 119  
TimeFinder/Snap operations ................................................... 120  
TimeFinder/Mirror operations .................................................. 122  
EMC Replication Manager ....................................................... 123  
EMC Storage Resource Management ...................................... 126  
EMC PowerPath ....................................................................... 130  
EMC Open Replicator ............................................................. 132  
EMC Virtual Provisioning .......................................................... 133  
Thin device ............................................................................... 133  
Data device ............................................................................... 133  
New Symmetrix VMAX Virtual Provisioning features ............ 134  
New Symmetrix VMAX TimeFinder/Clone features ............... 135  
EMC Fully Automated Storage Tiering (FAST) ......................... 137  
FAST VP .................................................................................. 138

# Chapter 3 DB2 for Linux, UNIX, and Windows Back Up and Recovery Concepts

Database recovery concepts ......................................................... 140  
Crash recovery ......................................................................... 140  
Version recovery ................................................................. 142  
Roll-forward recovery .......................................................... 143  
Recoverable and nonrecoverable databases ............................ 144  
Online versus offline back up and recovery ............................ 146  
Incremental and delta back up and recovery ............................ 146
Performing a crash recovery operation .................................................. 148
A word about soft checkpoints ...................................................... 151
Back up and recovery ........................................................................ 152
The DB2 Backup utility ..................................................................... 152
The DB2 Restore utility ................................................................. 158
The DB2 Roll-forward utility .......................................................... 163
A word about the recovery history file ........................................ 169
The DB2 Recover utility ................................................................. 171
Rebuilding invalid indexes .............................................................. 173
Backing up a database with split mirroring .................................... 174
Initializing a split mirror with db2inidb ............................................ 176

Chapter 4 EMC TimeFinder Family of Products

Introduction ..................................................................................... 180
Symmetrix device types used in TimeFinder operations ......... 183
Standard device ................................................................. 183
Business continuation volume (BCV) device ......................... 183
Clone device ................................................................. 184
Virtual device (VDEV) .................................................... 185
SAVE device (SAVEDEV) .............................................. 185
Metadevice ................................................................. 186
DATA device ............................................................... 186
Thin device ................................................................. 187
Performing TimeFinder/Clone operations ................................. 188
Clone copy sessions ................................................................. 188
Performing TimeFinder/Snap operations .................................... 190
Snap copy sessions ................................................................. 191
Performing TimeFinder/Mirror operations .................................. 193
TimeFinder/Mirror Establish operations ................................ .... 193
TimeFinder/Mirror split operations ........................................... 195
TimeFinder/Mirror restore operations ...................................... 196
TimeFinder consistent split ...................................................... 197
Enginuity Consistency Assist ...................................................... 197
TimeFinder/Mirror reverse split .............................................. 200

Chapter 5 Backing Up a DB2 LUW Database Using TimeFinder Technology

Using disk-based replication to back up a DB2 LUW database .................................................. 203
Planning for disk-based replication recovery ............................ 204
Backing up a DB2 LUW database that has been taken offline . 207
Creating an offline database back up image with
TimeFinder/Clone ................................................................. 207
Creating an offline database back up image with
TimeFinder/Snap .................................................................. 213
Backing up a DB2 LUW database while it remains online ...... 220
Creating an online database back up image with
TimeFinder/Clone ................................................................... 221
Creating an online database back up image with
TimeFinder/Snap ..................................................................... 226
Backing up the database copy ................................................. 231

Chapter 6  Restoring a DB2 LUW Database Using TimeFinder Technology

Using disk-based replication to restore a DB2 LUW database 235
Restoring a nonrecoverable DB2 LUW database from an offline database backup copy .................................................... 237
  Restoring a nonrecoverable database from an offline database back up image that was created with
  TimeFinder/Clone ...................................................................... 237
  Restoring a nonrecoverable database from an offline database back up image that was created with
  TimeFinder/Snap ..................................................................... 244
Restoring a nonrecoverable DB2 LUW database from an online database copy ................................................................. 252
  Restoring a nonrecoverable database from an online database back up image that was created with
  TimeFinder/Clone ...................................................................... 252
  Restoring a nonrecoverable database from an online database back up image that was created with
  TimeFinder/Snap ..................................................................... 260
Restoring a recoverable DB2 LUW database from an offline database copy ................................................................. 269
  Restoring a recoverable database from an offline database back up image that was created with
  TimeFinder/Clone ...................................................................... 269
  Restoring a recoverable database from an offline database back up image that was created with
  TimeFinder/Snap ..................................................................... 278
Restoring a recoverable DB2 LUW database from an online database copy ................................................................. 287
  Restoring a recoverable database from an online database back up image that was created with
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 DB2 family editions ....................................................................</td>
<td>28</td>
</tr>
<tr>
<td>2 The Control Center (advanced view) ...........................................</td>
<td>31</td>
</tr>
<tr>
<td>3 The Control Center toolbar .......................................................</td>
<td>32</td>
</tr>
<tr>
<td>4 The Command Editor .......................................................................</td>
<td>34</td>
</tr>
<tr>
<td>5 The Configuration Assistant .......................................................</td>
<td>36</td>
</tr>
<tr>
<td>6 The Command Line Processor (in interactive input mode) ...............</td>
<td>38</td>
</tr>
<tr>
<td>7 Hierarchical relationship between systems, instances, and databases</td>
<td>40</td>
</tr>
<tr>
<td>8 Invoking the Create Database Wizard from the Control Center .......</td>
<td>45</td>
</tr>
<tr>
<td>9 The first page of the Create Database Wizard .............................</td>
<td>46</td>
</tr>
<tr>
<td>10 Typical directory hierarchy tree for a nonpartitioned database ......</td>
<td>48</td>
</tr>
<tr>
<td>11 How data is written to table space containers ...........................</td>
<td>54</td>
</tr>
<tr>
<td>12 Invoking the Create Table Space Wizard from the Control Center ....</td>
<td>58</td>
</tr>
<tr>
<td>13 The first page of the Create Table Space Wizard .........................</td>
<td>59</td>
</tr>
<tr>
<td>14 Invoking the Alter Table Space dialog box from the Control Center ..</td>
<td>60</td>
</tr>
<tr>
<td>15 The first page of the Alter Table Space dialog box .....................</td>
<td>61</td>
</tr>
<tr>
<td>16 The transaction logging process .................................................</td>
<td>65</td>
</tr>
<tr>
<td>17 Circular logging ........................................................................</td>
<td>67</td>
</tr>
<tr>
<td>18 Archival logging ........................................................................</td>
<td>69</td>
</tr>
<tr>
<td>19 Invoking the DB2 Registry management tool from the Configuration Assistant .........................................................</td>
<td>77</td>
</tr>
<tr>
<td>20 DB2 Registry management tool dialog box ......................................</td>
<td>78</td>
</tr>
<tr>
<td>21 Invoking the DBM Configuration dialog box from the Control Center ...............................................................................</td>
<td>81</td>
</tr>
<tr>
<td>22 DBM Configuration dialog box ......................................................</td>
<td>82</td>
</tr>
<tr>
<td>23 Invoking the Database Configuration dialog box from the Control Center ...............................................................................</td>
<td>85</td>
</tr>
<tr>
<td>24 Database Configuration dialog box ..............................................</td>
<td>86</td>
</tr>
<tr>
<td>25 EMC Symmetrix VMAX Series with Enginuity ..................................</td>
<td>92</td>
</tr>
<tr>
<td>26 Basic synchronous SRDF configuration .........................................</td>
<td>100</td>
</tr>
<tr>
<td>27 SRDF consistency group ..................................................................</td>
<td>103</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>28</td>
<td>SRDF establish and restore control operations</td>
</tr>
<tr>
<td>29</td>
<td>SRDF failover and fallback control operations</td>
</tr>
<tr>
<td>30</td>
<td>Geographically distributed four-node EMC SRDF/CE clusters</td>
</tr>
<tr>
<td>31</td>
<td>EMC Symmetrix configured with standard volumes and BCVs</td>
</tr>
<tr>
<td>32</td>
<td>A TimeFinder/Clone copy of a standard device</td>
</tr>
<tr>
<td>33</td>
<td>Copy of a standard device to a virtual device (VDEV)</td>
</tr>
<tr>
<td>34</td>
<td>SRM commands</td>
</tr>
<tr>
<td>35</td>
<td>Virtual Provisioning components</td>
</tr>
<tr>
<td>36</td>
<td>Crash recovery</td>
</tr>
<tr>
<td>37</td>
<td>Version recovery</td>
</tr>
<tr>
<td>38</td>
<td>Roll-forward recovery</td>
</tr>
<tr>
<td>39</td>
<td>Initiating a crash recovery operation from the Control Center</td>
</tr>
<tr>
<td>40</td>
<td>Invoking the Backup Wizard from the Control Center</td>
</tr>
<tr>
<td>41</td>
<td>The first page of the Backup Wizard</td>
</tr>
<tr>
<td>42</td>
<td>Invoking the Restore Data Wizard from the Control Center</td>
</tr>
<tr>
<td>43</td>
<td>The first page of the Restore Database Wizard</td>
</tr>
<tr>
<td>44</td>
<td>Invoking the Roll-forward Wizard from the Control Center</td>
</tr>
<tr>
<td>45</td>
<td>The first page of the Roll-forward Wizard</td>
</tr>
<tr>
<td>46</td>
<td>Roll-forward page of the Restore Data Wizard</td>
</tr>
<tr>
<td>47</td>
<td>Creating a copy session using the symclone command</td>
</tr>
<tr>
<td>48</td>
<td>TimeFinder/Snap copy of a standard device to a VDEV</td>
</tr>
<tr>
<td>49</td>
<td>ECA consistent split across multiple database-associated hosts</td>
</tr>
<tr>
<td>50</td>
<td>ECA consistent split on a local Symmetrix system</td>
</tr>
<tr>
<td>51</td>
<td>Configuration to facilitate TimeFinder backup and recovery</td>
</tr>
</tbody>
</table>
## Tables

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Differences between SMS and DMS/AS table spaces</td>
<td>56</td>
</tr>
<tr>
<td>2 db2set command options</td>
<td>75</td>
</tr>
<tr>
<td>3 SYMCLI base commands</td>
<td>95</td>
</tr>
<tr>
<td>4 Data object SRM commands</td>
<td>127</td>
</tr>
<tr>
<td>5 Data object mapping commands</td>
<td>128</td>
</tr>
<tr>
<td>6 File system SRM commands to examine file system mapping</td>
<td>128</td>
</tr>
<tr>
<td>7 File system SRM command to examine logical volume mapping</td>
<td>129</td>
</tr>
<tr>
<td>8 SRM statistics command</td>
<td>129</td>
</tr>
<tr>
<td>9 Differences between recoverable and nonrecoverable databases</td>
<td>144</td>
</tr>
<tr>
<td>10 DB2 back up image file name components</td>
<td>157</td>
</tr>
<tr>
<td>11 TimeFinder device type summary</td>
<td>190</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 TechBook 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 the product release notes.

This TechBook provides a high-level overview of DB2 for Linux, UNIX, and Windows (LUW) and a general description of EMC products and utilities that can be used to store and manage DB2 LUW databases. It also provides information on how to use EMC’s TimeFinder technology to back up and restore a DB2 LUW database that has been deployed on an EMC Symmetrix storage system. While much of the content presented focuses on single-partition databases, considerations for multipartition databases are also addressed.

Audience

This TechBook is written primarily for IT professionals who have some experience working with DB2 for Linux, UNIX, and Windows (LUW), and are planning on developing a backup and recovery strategy using EMC’s TimeFinder technology in conjunction with a Symmetrix storage system. However, anyone who would like to learn how TimeFinder can be used to replicate DB2 LUW databases stored on Symmetrix storage systems will benefit from the information found in this book.

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

- Symmetrix operating environments
- DB2 for Linux, UNIX, and Windows operations and concepts
Related documentation

Other related EMC publications include:

- Symmetrix VMAX Series Product Guide
- Symmetrix DMX-4 Product Guide
- Symmetrix Remote Data Facility (SRDF) Product Guide
- EMC Solutions Enabler Symmetrix SRDF CLI Product Guide
- EMC Solutions Enabler Symmetrix TimeFinder CLI Product Guide
- Enginuity - The EMC Symmetrix Storage Operating Environment - A Detailed Review, White Paper
- EMC Symmetrix VMAX Series Update for the Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases TechBook, Technical Note
- New Features in EMC Enginuity 5874 for Symmetrix Open Systems Environments, White Paper
- EMC Symmetrix Enginuity Release Notes (multiple release levels available)
- EMC Solutions Enabler Version 7.0 Release Notes

Other related IBM publications include:

- IBM DB2 Database Administration Concepts and Configuration Reference
- IBM DB2 Data Recovery and High Availability Guide and Reference

For more IBM DB2 reference material, refer to:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp
http://www.ibm.com/redbooks

Author’s publications include:

- DB2 9 Fundamentals: Certification Study Guide
- DB2 9 for Linux, UNIX, and Windows Database Administration: Certification Study Guide
- Deploying DB2 for Linux, UNIX, and Windows Databases on EMC Symmetrix Arrays
- Using EMC TimeFinder to Clone DB2 for Linux, UNIX, and Windows Databases TechBook
- Using EMC SRDF to Facilitate Disaster Recovery for DB2 for Linux, UNIX, and Windows Databases
Conventions used in this document

EMC uses the following conventions for special notices.

---

**Note**: A note presents information that is important, but not hazard-related.

---

**IMPORTANT**

An important notice contains information essential to operation of the software or hardware.

---

**Typographical conventions**

EMC uses the following typographical conventions in this TechBook:

- **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
DB2 command/SQL statement syntax conventions
Many examples of DB2 LUW administrative commands and SQL statements can be found throughout this book. The following conventions are used whenever a DB2 command or SQL statement is presented:

- [ ] Parameters or items shown inside of brackets are required and must be provided.
- < > Parameters or items shown inside of angle brackets are optional and do not have to be provided.
- | Vertical bars are used to indicate that one (and only one) item in the list of items presented can be specified.
- , . . . A comma followed by three periods (ellipsis) indicate that multiple instances of the preceding parameter or item can be included in the DB2 command or SQL statement.

The following examples illustrate each of these conventions:

Example 1
```
REFRESH TABLE [TableName ,...]
<INCREMENTAL | NON INCREMENTAL>
```
In this example, at least one TableName value must be provided, as indicated by the brackets ([ ]), and more than one TableName value can be provided, as indicated by the comma-ellipsis (, . . . ) characters that follow the TableName parameter. INCREMENTAL and NON INCREMENTAL are optional, as indicated by the angle brackets (< >), and either one or the other can be specified, but not both, as indicated by the vertical bar (|).

Example 2
```
CREATE SEQUENCE [SequenceName]
<AS [SMALLINT | INTEGER | BIGINT | DECIMAL]>
<START WITH [StartingNumber]>
<INCREMENT BY [1 | Increment]>
<NO MINVALUE | MINVALUE [MinValue]>
<NO MAXVALUE | MAXVALUE [MaxValue]>
<NO CYCLE | CYCLE>
<NO CACHE CACHE 20 | CACHE [CacheValue]>
<NO ORDER | ORDER>
```
In this example, a SequenceName value must be provided, as indicated by the brackets ([ ]). However, everything else is optional, as indicated by the angle brackets (< >), and in many cases, a list of available option values is provided (for example, NO CYCLE and CYCLE); however, only one can be specified, as indicated by the vertical bar (|). In addition, when some options are provided (for
example, \texttt{START WITH, INCREMENT BY, MINVALUE, MAXVALUE, and CACHE}), a corresponding value must be provided, as indicated by the brackets ([ ])) that follow the option.

\textbf{Note:} SQL is not a case-sensitive language, but for clarity, the examples provided are shown in mixed case; command syntax is presented in uppercase while user-supplied elements such as table names and column names are presented in lowercase. However, the examples shown can be entered in any case.

\textbf{The team that wrote this TechBook}

This TechBook was authored by a team of engineers from EMC and a DB2 expert at IBM’s Toronto lab.

\begin{itemize}
  \item \textbf{Roger E. Sanders} is a Consultant Corporate Systems Engineer in the Integrated Customer Operations team at EMC. He has nine years of experience in the storage industry and he has been working with DB2 for Linux, UNIX, and Windows since it was first introduced on the IBM PC (as part of OS/2 1.3 Extended Edition). Roger has written articles for IDUG Solutions Journal and Certification Magazine, authored DB2 tutorials for IBM’s developerWorks website, presented at several International DB2 User's Group (IDUG) and regional DB2 User's Group (RUG) conferences, taught numerous classes on DB2 Family Fundamentals and DB2 Database Administration (DB2 for Linux, UNIX, and Windows), and is the author of 20 books on DB2 for Linux, UNIX, and Windows and one book on ODBC. For the past eight years, Roger has authored the Distributed DBA column in IBM Data Management Magazine (formerly DB2 Magazine) and he has helped IBM develop 17 DB2 Certification Exams. In 2008 and 2010, Roger was recognized as an IBM Information Champion; in 2010, he was recognized as an IBM developerWorks Contributing Author.

  \item \textbf{Paul Pendle} is a Consulting Systems Integration Engineer in the EMC Database and Applications Team. A 10-plus year veteran with EMC, Paul coordinates the technical activities for IBM Toronto labs and EMC to foster integration and compatibility. In the 20 years prior to EMC, Paul worked on database interaction with storage systems on both mainframe and distributed systems for several software and hardware companies, with an emphasis on physical and logical design of data models for optimal performance. In 2010, Paul was recognized as an IBM Information Champion.
\end{itemize}
◆ Dale McInnis is a Senior Technical Staff Member (STSM) at the IBM Toronto lab where DB2 for Linux, Unix, and Windows is developed. He has a B.Sc.(CS) from the University of New Brunswick and a Masters of Engineering from the University of Toronto. Dale joined IBM in 1988, and has been working on the DB2 development team since 1992. During this time, Dale has always focused on the DB2 Kernel development, where he led the team that designed the current back up and recovery architecture, and the integration of datalinks technology. Dale is currently the DB2 High Availability Architect and is known in the industry as the expert in this field. He is focused on the availability features and functions for DB2 LUW’s future releases and its integration with other software from IBM such as Tivoli System Automation and Tivoli Storage Manager.

Additional contributors to this book:

◆ Geraldine Ledoux, Integrated Customer Operations, EMC Corporation, Hopkinton, MA
◆ Jim Wentworth, Integrated Customer Operations, EMC Corporation, Hopkinton, MA
◆ Chris Fallon, Integrated Customer Operations, EMC Corporation, Hopkinton, MA
◆ Don Fried-Tanzer, Integrated Customer Operations, EMC Corporation, Hopkinton, MA
◆ Paul Pendle, Integrated Customer Operations, EMC Corporation, Hopkinton, MA

We’d like to hear from you!

Your feedback on our TechBooks is important to us! We want our books to be as helpful and relevant as possible, so please feel free to send us your comments, opinions and thoughts on this or any other TechBook:

TechBooks@emc.com
DB2 version 9.7 for Linux, UNIX, and Windows (LUW) and DB2 pureScale (which technically, is version 9.8), are the latest releases of IBM’s open-systems hybrid database management system. In addition to the functionality introduced with DB2 version 9, version 9.7 delivers important new features and enhancements that address business needs, whether those needs are integrating business data from across your organization, reducing costs, creating business value, or providing a secure and resilient system for your company’s valuable information assets. DB2 pureScale builds on version 9.7 by providing an active-active shared-disk database implementation that is based on the DB2 for z/OS data sharing architecture.

This chapter is designed to introduce you to the various products that make up the core of the DB2 Family and to the more common set of tools that are available to assist in the administration and management of DB2 servers, instances, databases, and database objects. This chapter is also designed to show you how to configure a server, an instance, or a database, to show you what happens when a new database is created, and to provide you with an overview of how data for a database is physically stored. Topics covered include:

- The DB2 Family ................................................................. 22
- DB2’s comprehensive tool set ........................................... 29
- Server, instances, and databases ...................................... 39
- Objects that make up a DB2 database environment ............ 42
- Creating a DB2 LUW database ......................................... 44
- A closer look at table spaces ............................................. 54
- A closer look at transaction logging .................................. 63
- Configuring a DB2 LUW database environment ............... 73
DB2, an acronym for DATABASE 2, was born on MVS in 1983. In 1987, DB2 arrived on the personal computer (PC) as the Database Manager in OS/2 1.3 Extended Edition and a year later, it emerged as SQL/400 for IBM’s new AS/400 server. By 1992, DB2 had become a stand-alone product on OS/2 (it now had the name DB2/2) and in 1993, DB2 appeared on AIX. (This prompted another name change and DB2/2 became DB2 for Common Servers.) New editions of DB2 were introduced on HP-UX and Solaris in 1994, on Windows in 1995, and on Linux in 1999. (Along the way, the name changed again and DB2 for Common Servers became DB2 Universal Database or DB2 UDB.)

DB2 version 9.7 for Linux, UNIX, and Windows and the DB2 pureScale Feature (which technically, is version 9.8), are the latest releases of IBM’s popular data management software for distributed open-systems. (Starting with version 9, the name changed again). Like previous versions, DB2 runs on a wide variety of platforms (AIX, HP-UX, Linux, Solaris, Windows, i5/OS, and z/OS), and several editions are available — each of which has been designed to meet a specific business need. These editions, along with an extensive suite of add-on products that provide additional storage capability and advanced connectivity, are collectively known as the DB2 Family. The editions that make up the heart of the DB2 Family are:

- **DB2 Everyplace.** DB2 Everyplace is a small footprint (approximately 350 KB) relational database and a high performance data synchronization solution that allows enterprise applications and data to be extended to mobile devices like personal digital assistants (PDAs), handheld personal computers (HPCs), and smart phones. DB2 Everyplace can be used as a local, stand-alone database that resides on a mobile device or to access information stored on remote servers whenever a connection is available. DB2 Everyplace can also be embedded directly into mobile devices to increase their functionality.

DB2 Everyplace is available in two editions: DB2 Everyplace Database Edition and DB2 Everyplace Enterprise Edition. DB2 Everyplace Database Edition is designed to be used by Independent Software Vendors (ISVs) and application developers who wish to create powerful mobile and embedded applications that work with DB2 Everyplace database data stored directly on a mobile device. DB2 Everyplace Enterprise Edition is designed to
be a complete datacentric mobile synchronization server. This secure server is responsible for managing the distribution and synchronization of data between mobile device users and back-end data sources, such as DB2, Informix, Oracle, Sybase, and Microsoft SQL Server. (Synchronization is performed whenever a connection to the back-end data source is detected.)

- **DB2 Express.** DB2 Express Edition (or DB2 Express) is an entry-level data server that is designed to be used on microcomputers that have up to two CPUs (a dual-core processor is treated as a single CPU), up to 4 GB of memory, and are running a supported version of Linux, Solaris, or Windows. DB2 Express contains a rich feature set that will meet the needs of most deployments; for workloads or environments that require additional functionality, add-on features are available for an additional licensing fee.

- **DB2 Express-C.** DB2 Express-C is a no-charge, entry-level, data server that is designed to be used on microcomputers that have up to two CPUs, up to 4 GB of memory, and are running a supported version of Linux or Windows. DB2 Express-C is intended to be used for evaluation purposes and for the development/deployment of C, C++, Java, .NET, PHP, and XQuery applications. Essentially, DB2 Express-C is a subset of DB2 Express Edition with one exception: where pureXML is available as an add-on feature for DB2 Express, it is included with DB2 Express-C.

- **DB2 Personal Edition.** DB2 Personal Edition (PE) is a single-user, full-function, relational database management system that is ideal for desktop or laptop-based deployments. Databases under its control can be managed remotely, making it the perfect edition for occasionally connected or remote office implementations that do not require multiuser capability. With DB2 Personal Edition, a user can create, manipulate, and administer any number of local databases; however, each database created must reside on a storage medium that is managed by the PC that the DB2 software has been installed on. Remote clients cannot access databases that are under DB2 Personal Edition’s control, but PCs running DB2 Personal Edition can act as remote clients and access data stored on other DB2 servers. DB2 Personal Edition can be deployed on any PC that is running Linux or Windows — however, you must acquire a separate license for each user that will access a database under its control.
Introduction to DB2 for Linux, UNIX, and Windows

- **DB2 Workgroup Server Edition.** DB2 Workgroup Server Edition (WSE) is a multiuser, full-function, client/server database management system designed to be used on microcomputers that have up to four CPUs, up to 16 GB of memory, and are running any of the following operating systems: AIX, HP-UX, Solaris, Linux (32-bit and 64-bit), Novell Enterprise Server, and Windows (32-bit and 64-bit).

DB2 Workgroup Server Edition includes all of the features of DB2 Express, while providing scalability to larger servers. Thus, it is the ideal data server for small- to medium-size business environments and departments that are comprised of a small number of internal users.

- **DB2 Enterprise Server Edition.** DB2 Enterprise Server Edition (ESE) is a multiuser, full-function, Web-enabled client/server database management system that easily scales to handle high-volume transaction processing, multiterabyte data warehouses, and mission-critical applications from vendors like SAP. It is designed to be used on any size server (from one to hundreds of CPUs) that is running any of the following operating systems: AIX, HP-UX, Solaris, Linux (32-bit and 64-bit), Novell Enterprise Server, and Windows (32-bit and 64-bit).

DB2 Enterprise Server Edition includes all of the functionality found in DB2 Workgroup Edition, plus features that are needed to handle high user loads and provide 24x7x365 availability, including:

- High Availability Disaster Recovery (HADR)
- Data partitioning
- Table (range) partitioning
- Online table and index reorganization
- Materialized Query Tables
- Multidimensional data clustering
- Full intra-query parallelism
- Connection Concentrator
- The DB2 Governor
- Tivoli System Automation for Multiplatforms (TSA MP)

DB2 Enterprise Server Edition also comes packaged with a tightly integrated connectivity product (DB2 Connect) that allows it to participate in heterogeneous networks using the Distributed
Relational Database Architecture (DRDA) protocol. This allows up to five users to interact with iSeries and zSeries-based DB2 databases, Informix Dynamic Server (IDS) databases, and nondatabase host resources like CICS, VSAM, and IMS. Designed for midsize to large businesses, DB2 Enterprise Server Edition is the ideal foundation for building multiterabyte data warehouses, high-availability, high-volume On-Line Transaction Processing (OLTP) systems, or Web-based Business Intelligence (BI) solutions.

- **DB2 pureScale.** DB2 pureScale is a feature for DB2 Enterprise Server Edition that builds on familiar and proven design features from the IBM DB2 for z/OS database software (DB2 for z/OS). It leverages proven technology from DB2 for z/OS to bring active-active shared-disk technology to open systems. The DB2 pureScale feature offers the following key benefits:

  - **Practically unlimited capacity** - DB2 pureScale provides practically unlimited capacity by allowing for the addition and removal of members on demand. DB2 pureScale can scale to 128 members and has a highly efficient centralized management facility that allows for very efficient scale-out capabilities. DB2 pureScale also leverages a technology called Remote Direct Memory Access (RDMA) that provides a highly efficient inter-node communication mechanism that also facilitates its scaling capabilities.

  - **Application transparency** - An application that runs in a DB2 pureScale environment does not need to have any knowledge of the different members in the cluster or to be concerned about partitioning data. DB2 pureScale will automatically route applications to the members deemed most appropriate. DB2 pureScale also provides native support for a great deal of syntax used by other database vendors, allowing those applications to run in a DB2 pureScale environment with minimal or no changes.

  - **Continuous availability** - DB2 pureScale provides a fully active-active configuration such that if one member goes down, processing can continue at the remaining active members. During a failure, only data being modified on the failing member is temporarily unavailable until database recovery completes for that set of data, which is very quick. This is in direct contrast to other competing solutions where an entire system freeze may occur as part of the database recovery process.
• **Reduced TCO** - DB2 pureScale can help reduce TCO through its integrated and simplified deployment and maintenance capabilities. The DB2 pureScale interfaces handle the deployment and maintenance of components integrated within the DB2 pureScale Feature. This helps reduce what might amount to steep learning curves that would be associated with some of the competing technologies.

- **DB2 Data Warehouse Edition.** DB2 Data Warehouse Edition (DWE) is the top-of-the-line DB2 Edition for dynamic data warehousing. It is designed for today’s data center environments where On-Line Transaction Processing (OLTP) and decision support are merged into integrated information management systems. This integrated platform for developing warehouse-based analytics includes core components for warehouse construction and administration as well as Web-based applications with embedded data mining and multidimensional Online Analytical Processing (OLAP).

  The core engine for DB2 Data Warehouse Edition is DB2 Enterprise Server Edition and the DB2 Data Partitioning Feature. (DB2 Enterprise Server Edition includes data warehouse enhancing features such as materialized query tables, the starburst optimizer, and multidimensional clusters; the DB2 Data Partitioning Feature provides increased parallelism to aid in performing administration tasks, as well as scalability to support very large databases and complex workloads.)

- **DB2 for i5/OS.** DB2 for i5/OS is an advanced, 64-bit relational database management system that leverages the On-Demand capabilities of System i, such as Dynamic Logical Partitioning, to quickly respond to changing workloads in order to ensure business continuity in a dynamic environment. Unlike other DB2 editions, DB2 for i5/OS is built directly into the operating system. As a result, Version/Release naming will differ because DB2 for i5/OS follows the i5/OS version/release numbering scheme, and not the DB2 for Linux, UNIX, and Windows version/release scheme. The current level of DB2 for i5/OS is Version 6 Release 1 (V6R1).

  DB2 for i5/OS’s cost-based query optimizer, unique single level store architecture, and database parallelism feature allow it to scale near-linearly within an iSeries’ SMP configuration. And if additional functionality is needed, there are several utilities available (including utilities for data replication, parallel
processing, and query management) that can either be added to the core database functionality or that are included in the System i Enterprise Edition bundle.

- **DB2 for z/OS.** DB2 for z/OS is a multiuser, full-function, database management system that has been designed specifically for z/OS, IBM’s flagship mainframe operating system. For over four decades the IBM mainframe has been a leader in data and transaction serving; DB2 9 for z/OS builds on the value delivered by the IBM mainframe. DB2 9 for z/OS is designed to significantly cut IT infrastructure costs, streamline efforts to meet compliance obligations, and simplify data serving on the System z9 operating system.
All of the DB2 Family editions available, along with the type of computing environment each edition is primarily designed for, can be seen in Figure 1.
DB2’s comprehensive tool set

With the exception of DB2 Everyplace, DB2 for i5/OS, and DB2 for z/OS, each edition of DB2 comes with a comprehensive set of tools designed to assist in administering and managing DB2 instances, databases, and database objects. The majority of these tools have a graphical user interface (GUI); however, most of the tasks that can be performed with the GUI tools available can also be performed by issuing equivalent DB2 commands from the operating system prompt, the DB2 Command Editor, or the DB2 Command Line Processor. Of this set, the tools that are used most often include:

- The Control Center
- The Command Editor
- The Configuration Assistant
- The Command Line Processor

IMPORTANT
The GUI tools available with version 9.7 have been deprecated and will be replaced with the Optim family of products in a future release.

The Control Center

Of all the DB2 GUI tools available, the Control Center is the most important and versatile. The Control Center presents a clear, concise view of an entire system and serves as the central point for managing DB2 systems and performing common administration tasks. With the Control Center, users can:

- Create and delete instances.
- Create and delete (drop) DB2 databases.
- Catalog and uncatalog databases.
- Configure instances and databases.
- Create, alter, and drop buffer pools, table spaces, tables, views, indexes, aliases, triggers, schemas, and user-defined data types (UDTs).
- Grant and revoke authorities and privileges.
Introduction to DB2 for Linux, UNIX, and Windows

- Export, import, or load data.
- Reorganize tables and collect table statistics.
- Back up and restore databases and table spaces.
- Replicate data between systems.
- Manage database connections.
- Monitor resources and track events as they take place.
- Analyze queries.
- Schedule jobs to run unattended.

The Control Center interface presents itself using one of three different views:

- **Basic.** The basic view displays essential objects such as databases, tables, views, and stored procedures and limits the actions you can perform to those objects. This is the view you should use if you only want to perform core DB2 database operations.

- **Advanced.** The advanced view displays all objects available in the Control Center and allows you to perform all actions available. This is the view you should use if you are working in an enterprise environment and/or if you want to connect to DB2 for i5/OS or DB2 for z/OS.

- **Custom.** The custom view gives you the ability to tailor the object tree and actions allowed to meet your specific needs.

*Figure 2 on page 31 shows how the Control Center looks on a Windows XP server when the advanced view is used.*
Introduction to DB2 for Linux, UNIX, and Windows

Figure 2 The Control Center (advanced view)

If you look closely at Figure 2, you will notice that the Control Center is comprised of the following elements:

- A **menu bar**, which allows users to perform any of the Control Center functions available.

- A **toolbar**, which can be used to launch the other DB2 GUI tools available. Figure 3 on page 32 identifies the tools that can be invoked directly from the Control Center toolbar.
It is important to note that every tool that can be invoked from the Control Center toolbar can also be invoked from the Control Center’s menu bar.

- An objects pane (located on the left-hand side of the Control Center), which contains a hierarchical representation of every object type that can be managed from the Control Center.
- A contents pane (located on the upper right-hand side of the Control Center), which contains a listing of existing objects that correspond to the object type selected in the objects pane. (For example, if the Tables object type were selected in the objects pane, a list of all tables available would be listed in the contents pane.)
- An objects details pane (located on the lower right-hand side of the Control Center), which contains detailed information about the object selected in the object tree or contents pane.
The Command Editor

The Command Editor is an interactive GUI application that is used to generate, edit, execute, and manipulate SQL statements and DB2 commands; to work with the resulting output; and to view a graphical representation of the access plan chosen for explained SQL statements. From the Command Editor, users can:

- Execute SQL statements, DB2 commands, and operating system commands — operating system commands must be preceded by an exclamation mark (!).
- View the results of the execution of SQL statements and DB2 commands and see the result data set produced in response to a query.
- Save the results of the execution of SQL statements and DB2 commands to an external file.
- Create and save a sequence of SQL statements and DB2 commands to a script file that can be invoked by the Task Center. (Such a script file can then be scheduled to run at a specific time or frequency.)
- Use the SQL Assist tool to build complex queries.
- Examine the execution plan and statistics associated with a SQL statement before (or after) it is executed.

Figure 4 on page 34 shows how the Command Editor looks on a Windows XP server after a database connection has been established.
Introduction to DB2 for Linux, UNIX, and Windows

Figure 4  The Command Editor

As Figure 4 shows, the Command Editor is comprised of three individual pages (which are accessed by tabs): the Commands page, the Query Results page, and the Access Plan page. Users can enter and execute an SQL statement or a DB2 command, create and save a script, run an existing script, or schedule a task from the Commands page. Once a query has been executed, users can see the results, if any, on the Query Results page. And on the Access Plan page, users can see the access plan for any explainable statement that was specified on the Commands page. (If more than one SQL statement is specified on the Commands page, an access plan will be created only for the first statement encountered.)
The Configuration Assistant

The Configuration Assistant is an interactive GUI application that allows users to configure clients so they can access databases stored on remote DB2 servers. In order to access an instance or database on another server/system, that system must first be cataloged in the node directory of the client workstation, and information about the remote database must be cataloged in the database directory (also on the client workstation). The Configuration Assistant provides a way to quickly catalog nodes and databases without having to know the inherent complexities involved with performing these tasks. And because the Configuration Assistant maintains a list of databases to which users/applications can connect, it can act as a lightweight alternative to the Control Center in situations where the complete set of GUI tools available has not been installed.

From the Configuration Assistant, users can:

◆ Catalog new databases.
◆ Work with or uncatalog existing databases.
◆ Bind applications.
◆ Set DB2 environment/registry variables.
◆ Configure the DB2 Database Manager instance.
◆ Configure ODBC/CLI parameters.
◆ Import and export configuration information.
◆ Change passwords.
◆ Test connections.

Figure 5 on page 36 shows how the Configuration Assistant might look on a Windows XP server.
The Command Line Processor

The Command Line Processor (CLP) is a text-oriented application that allows users to issue DB2 commands, system commands, and SQL statements, as well as view the results of the statements/commands executed. The Command Line Processor can be run in three modes:
◆ **Command mode.** When the Command Line Processor is run in command mode, the user simply enters a DB2 command or SQL statement, preceded by the characters “db2”, at the system prompt. (For example, the command `CONNECT TO sample` would be entered as `db2 CONNECT TO sample`.) If the command contains characters that have a special meaning to the operating system being used, it must be enclosed in quotation marks to ensure that it will be properly executed (for example, `db2 "SELECT COUNT(*) FROM employee"`). If the command to be executed is too long to fit on a single line, a space followed by the line continuation character (\) can be placed at the end of the line that is to be continued, and the rest of the command can follow on a new line.

◆ **Interactive Input mode.** When the Command Line Processor is run in interactive input mode, the “db2” prefix is automatically provided (as characterized by the `db2 =>` input prompt) for each command/SQL statement entered. To run the Command Line Processor in interactive input mode, you simply enter the command “db2” at the system prompt. To exit out of interactive, you enter the command “quit” at the Command Line Processor prompt. Aside from that, the rules that apply to using the command mode of the Command Line Processor also apply to using the interactive input mode.

◆ **Batch mode.** When the Command Line Processor is run in batch mode, it is assumed that all commands and/or SQL statements to be executed have been stored in an ASCII-format text file. (The characters “db2” should not precede the commands/statements stored in this file.) To run the Command Line Processor in batch mode, you simply enter the command `db2 -f x x x x x x x x x x` (where `x x x x x x x x x x` is the name of the file that contains the set of commands that are to be executed) at the system prompt.

Figure 6 on page 38 shows how the Command Line Processor looks on a Windows XP server when it is run in interactive input mode.
There are various command-line options that can be specified when the Command Line Processor is invoked; a list of all options available can be obtained by executing the command **LIST COMMAND OPTIONS**, either from the system prompt or the Command Line Processor prompt (when the Command Line Processor is ran in interactive input mode).
Introduction to DB2 for Linux, UNIX, and Windows

Server, instances, and databases

DB2 for Linux, UNIX, and Windows sees the world as a hierarchy of objects. Workstations (or servers) on which DB2 has been installed occupy the highest level of this hierarchy. During the installation process, program files for a background process known as the DB2 Database Manager are physically copied to a specific location on the server and an instance of the DB2 Database Manager is created. (The default instance for a particular system is defined by the DB2INSTANCE environment variable and this is the instance used for most operations.)

Instances occupy the second level in the hierarchy and are responsible for managing system resources and databases that fall under their control. Although only one instance is created initially, several instances can exist. Each instance behaves like a separate installation of DB2, even though all instances within a system share the same DB2 Database Manager program files (unless each instance is running a different version of DB2). And although multiple instances share the same binary code, each runs independently of the others and has its own environment, which can be modified by altering the contents of its associated configuration file.

Every instance controls access to one or more databases; databases make up the third level in the hierarchy and are responsible for managing the storage, modification, and retrieval of data. Like instances, databases work independently of each other. Each database has its own environment (also controlled by a set of configuration parameters), as well as its own set of grantable authorities and privileges to govern how users interact with the data and database objects it controls. From a user’s perspective, a database is a collection of tables (preferably related in some way) that are used to store data. However, from a database administrator’s viewpoint, a DB2 LUW database is much more; a database is an entity that is comprised of many physical and logical components. Some of these components help determine how data is organized, while others determine how and where data is physically stored. Figure 7 on page 40 shows the hierarchical relationship between systems, instances, and databases.
Introduction to DB2 for Linux, UNIX, and Windows

The DB2 Administration Server (DAS) instance

The tools that come with DB2, such as the Control Center and the Command Center, require a separate instance that operates independently of, yet concurrently with, all other instances that have been defined for a particular workstation. For this reason, a special instance, known as the DB2 Administration Server (DAS) instance, is also created as part of the DB2 installation process. In contrast to other instances, only one DAS instance can exist on a single workstation. (The DB2 global-level profile registry variable DB2ADMINSERVER contains the name of the DAS instance that has been defined for a particular workstation.)
Once created, the DAS instance runs continuously as a background process whenever the system it was created on is online; the DAS instance is usually started automatically each time the workstation it resides on is booted. The DAS instance must be running on every DB2 server that you wish to administer remotely. That’s because, among other things, the DAS instance provides remote clients with the information needed to establish communications with other instances.

Note: To administer a server from a remote client, a user must have System Administration (SYSADM) authority for the DAS instance used. Furthermore, once a remote instance and database have been registered on a client workstation, the user must hold the authorities and privileges needed to perform administrative tasks.

In addition to enabling remote administration of DB2 servers, the DAS instance assists the Control Center and the Configuration Assistant in:

- Providing job (task) management, including the ability to schedule and run user-defined shell scripts/batch files that contain both DB2 and operating system commands.
- Scheduling jobs, viewing the results of completed jobs, and performing administrative tasks against jobs executed either remotely or locally (by using the Task Center).
- Providing a means for discovering information about the configuration of other DAS instances, DB2 instances, and databases using DB2 Discovery. (The Configuration Assistant and the Control Center use such information to simplify and automate the configuration of client connections to DB2 servers; neither tool will be able to “discover” a server if the DAS instance for that server is not running.)

IMPORTANT

The DB2 Administration Server (DAS) has been deprecated in version 9.7 and may be removed in a future release.
Objects that make up a DB2 database environment

DB2 LUW uses both a logical and a physical storage model that is comprised of several different, yet related, objects. Four types of objects exist. They are:

- **System objects**. System objects consist of registry variables, instance configuration files, and individual database configuration files. Registry variables are set at the system level and can affect every instance that resides on a particular server. Instance configuration files (also known as DB2 Database Manager configuration files) are created and assigned to individual instances during the instance creation process. Values in an instance’s configuration file control how resources are allocated for that particular instance, and changes to them affect every database that falls under that instance’s control. Similarly, database configuration files are created and assigned to individual databases during the database creation process. Values in a database’s configuration file control how resources are allocated for that particular database and changes to them can impact performance and control resource utilization.

- **Recovery objects**. Recovery objects consist of transaction log files and recovery history files. By default, one recovery history file and three transaction log files are automatically created when a database is created. Recovery log files are used, together with database back up images and transaction log files, to coordinate database recovery operations. The recovery history file contains information about every back up operation executed, while transaction log files contain records of recent database operations performed. In the event a database has to be recovered from an application, user, or system error, events stored in the transaction log files can be replayed to return the database to a consistent and stable state, or to return a database to the state it was in up to the point in time that the error occurred – provided roll-forward recovery is enabled. You cannot modify transaction log files or recovery history files directly; however, you can control where transaction log files are physically stored.

- **Storage objects**. Storage objects control where data is physically stored and how data is moved between storage and memory during normal operation. Three types of storage objects are used. They are:
• **Buffer pools.** A buffer pool is a section of memory that has been reserved for the sole purpose of caching data pages as they are read from physical storage. Whenever data is needed to resolve a query, the page the data is stored on is located in physical storage and transferred to a buffer pool, where it is then read and/or modified. If the page is modified, eventually it is copied back to physical storage; however, all pages read stay in memory until the space they occupy is needed or until all connections to the database are terminated. Furthermore, whenever a page of data is retrieved, the DB2 Database Manager uses a set of heuristic algorithms to try to determine which pages will be needed next and those pages are retrieved as well (this is referred to as prefetching). Page retention and prefetching are done to improve overall performance; data can be accessed much faster when it is stored in memory than when it is stored on disk.

• **Containers.** A container is some form of physical storage that the DB2 Database manager has reserved access to. A container can be a directory that may or may not already exist, a fixed-size, preallocated file that may or may not already exist, or a physical (raw) device that is recognized by the operating system. (On Linux and UNIX operating systems, a physical device can be any logical volume that uses a character special interface; on Windows operating systems, a physical device is any unformatted partition or any physical disk.)

• **Table spaces.** Table spaces are used to control where data is physically stored and to provide a layer of indirection between database objects (such as tables, indexes, and views) and one or more containers (that is, directories, files, or raw devices) in which the object’s data actually resides. A single table space can span many containers, but each container can only belong to one table space.

• **Data (or database) objects.** Data objects — otherwise known as database objects — are used to logically store and manipulate data, as well as to control how all user data (and some system data) is organized. Data objects include tables, indexes, views, aliases, schemas, triggers, user-defined data types, user-defined functions, and sequences.
Creating a DB2 LUW database

There are several ways to create a DB2 LUW database; the most common way is by using the Create Database Wizard or by executing the `CREATE DATABASE` command. In its simplest form, the syntax for the `CREATE DATABASE` command is:

```
CREATE [DATABASE | DB] [DatabaseName]
```

where:
- `DatabaseName` — Identifies a unique name that is to be assigned to the database once it is created.

The only value you must provide when executing this command is a name to assign to the new database. This name:
- Can consist of only the characters `a` through `z`, `A` through `Z`, `0` through `9`, `@`, `#`, `$`, and `_` (underscore);
- Cannot begin with a number;
- Cannot begin with the letter sequences “SYS,” “DBM,” or “IBM”; and
- Cannot be the same as the name already assigned to another database within the same instance.

When this form of the `CREATE DATABASE` command is executed, the characteristics of the database created, such as the storage location and transaction logging method used, are determined by several predefined defaults. If you wish to change any of the default characteristics, you must use a more complex form of the `CREATE DATABASE` command.

If you prefer using graphical user interfaces to typing long commands, you can use the Create Database Wizard to construct a DB2 LUW database. (The Create Database Wizard is designed to collect information that defines the characteristics of a database — and then create a database that has those characteristics. These same characteristics can be specified through the various options that are available with the `CREATE DATABASE` command.) The Create Database Wizard is invoked by selecting the appropriate action from the Databases menu found in the Control Center. Figure 8 on page 45 shows the Control Center menu items that must be selected to activate the Create Database Wizard; Figure 9 on page 46 shows what the first page of the Create Database Wizard looks like when it is first initiated.
Figure 8  Invoking the Create Database Wizard from the Control Center
Introduction to DB2 for Linux, UNIX, and Windows

Once the Create Database Wizard is displayed, you simply follow the directions shown on each panel presented to define the characteristics of the database that is to be created. When you have provided enough information for the DB2 Database Manager to create a database, the Finish button displayed in the lower-right corner of the wizard (see Figure 9) will be enabled. Once this button is selected, a database will be created using the information provided.

What happens when a DB2 LUW database is created

Regardless of how the process is initiated, whenever a new DB2 database (other than a DB2 pureScale database) is created, the following tasks are performed, in the order shown:

Figure 9  The first page of the Create Database Wizard

Once the Create Database Wizard is displayed, you simply follow the directions shown on each panel presented to define the characteristics of the database that is to be created. When you have provided enough information for the DB2 Database Manager to create a database, the Finish button displayed in the lower-right corner of the wizard (see Figure 9) will be enabled. Once this button is selected, a database will be created using the information provided.

What happens when a DB2 LUW database is created

Regardless of how the process is initiated, whenever a new DB2 database (other than a DB2 pureScale database) is created, the following tasks are performed, in the order shown:
1. All directories and subdirectories needed are created in the appropriate location.

Information about every DB2 database created is stored in a special hierarchical directory tree. Where this directory tree is actually created is determined by information provided with the `CREATE DATABASE` command — if no location information is provided, this directory tree is created in the location specified by the `dftdbpath` DB2 Database Manager configuration parameter associated with the instance under which the database is being created. The root directory of this hierarchical tree is assigned the name of the instance with which the database is associated. This directory will contain a subdirectory that has been assigned a name corresponding to the partition’s node. If the database is a partitioned database, this directory will be named `NODExxxx`, where `xxxx` is the unique node number that has been assigned to the partition; if the database is a nonpartitioned database, this directory will be named `NODE0000`. The node-name directory, in turn, will contain one subdirectory for each database that has been created, along with one subdirectory that includes the containers that are used to hold the database’s data.

The name assigned to the subdirectory that holds the containers used to house the database’s data is the same as that specified for the database; the name assigned to the subdirectory that contains the base files for the database corresponds to the database token that is assigned to the database during the creation process (the subdirectory for the first database created will be named `SQL00001`, the subdirectory for the second database will be named `SQL00002`, and so on). Figure 10 on page 48 illustrates how this directory hierarchy typically looks in a nonpartitioned database environment.
**Introduction to DB2 for Linux, UNIX, and Windows**

**Figure 10** Typical directory hierarchy tree for a nonpartitioned database

**IMPORTANT**

Never attempt to modify this directory structure or any of the files stored in it. Such actions could destroy one or more databases or make them unusable.
Files needed for management, monitoring, and database recovery are created.

After the subdirectory that was assigned the name of the database’s token is created, the following files are created in it:

- **db2rhist.asc** — This file contains historical information about back up operations, restore operations, table load operations, table reorganization operations, table space alterations, and similar database changes (i.e., the recovery history file).
- **db2rhist.bak** — This file is a back up copy of db2rhist.asc.
- **SQLBP.1** — This file contains buffer pool information.
- **SQLBP.2** — This file is a back up copy of SQLBP.1.
- **SQLDBCON** — This file contains database configuration information.
- **SQLDBCONF** — This file is a back up copy of SQLDBCON.
- **SQLINSLK** — This file contains information that is used to ensure that the database is assigned to only one instance of the DB2 Database Manager.
- **SQLLOGCTL.LFH** — This file contains information about active transaction log files. Recovery operations use information stored in this file to determine how far back in the logs to begin the recovery process.
- **SQLLOGMIR.LFH** — This file is a mirrored copy of SQLLOGCTL.LFH.
- **SQLSGF.1** — This file contains storage path information associated with automatic storage.
- **SQLSGF.2** — This file is a back up copy of SQLSGF.1.
- **SQLSPCS.1** — This file contains table space information.
- **SQLSPCS.2** — This file is a back up copy of SQLSPCS.1.
- **SQLTMPLK** — This file contains information about temporary table spaces.

Two subdirectories named **DB2EVENT** and **SQLLOGDIR** are also created; a detailed deadlocks event monitor is created and stored in the **DB2EVENT** subdirectory, and three files named **S0000000.LOG**, **S0000001.LOG**, and **S0000002.LOG** are created and stored in the **SQLLOGDIR** subdirectory. These three files are used to store transaction log records as SQL operations are performed against the database.
3. A buffer pool is created for the database.

   During the database creation process, a buffer pool is created and assigned the name **IBMDEFAULTBP**. By default, on Linux and UNIX platforms, this buffer pool is 1,000 4 KB (kilobyte) pages in size; on Windows platforms, this buffer pool is 250 4 KB pages in size. The actual memory used by this buffer pool (and for that matter, by any other buffer pools that may exist) is allocated when the first connection to the database is established and freed when all connections to the database have been terminated.

4. Two regular table spaces and one system temporary table space are created.

   Immediately after the buffer pool IBMDEFAULTBP is created, three table spaces are created and associated with this buffer pool. These three table spaces are as follows:
   
   - A regular table space named **SYSCATSPACE**, which is used to store the system catalog tables and views associated with the database
   - A regular table space named **USERSPACE1**, which is used to store all user-defined objects (such as tables, indexes, and so on) along with user data, index data, and long value data
   - A system temporary table space named **TEMPSPACE1**, which is used as a temporary storage area for operations such as sorting data, reorganizing tables, and creating indexes

   Unless otherwise specified, SYSCATSPACE and USERSPACE1 will be DMS FILE table spaces, and TEMPSPACE1 will be an SMS table space; characteristics for each of these table spaces can be provided as input to the **CREATE DATABASE** command or the Create Database Wizard.

5. The system catalog tables and views are created.

   After the table space SYSCATSPACE is created, a special set of tables, known as the system catalog tables, are constructed within that table space. The DB2 Database Manager uses the system catalog tables to keep track of information such as database object definitions, database object dependencies, database object privileges, column data types, table constraints, and object relationships. A set of system catalog views is created along with the system catalog tables, and these views are typically used when accessing data stored in the system catalog tables. The system catalog tables and views cannot be modified with SQL.
statements (however, their contents can be viewed). Instead, they are modified by the DB2 Database Manager whenever one of the following events occurs:

- A database object (such as a table, view, or index) is created, altered, or dropped.
- Authorizations or privileges are granted or revoked.
- Statistical information is collected for a table.
- Packages are bound to the database.

In most cases, the complete characteristics of a database object are stored in one or more system catalog tables when the object is created. However, in some cases, such as when triggers and constraints are defined, the actual SQL used to create the object is stored instead.

6. The database is cataloged in the system and local database directory (a system or local database directory is created first if it does not already exist).

DB2 uses a set of special files to keep track of where databases are stored and to provide access to those databases. Because the information stored in these files is used like the information stored in an office-building directory is used, they are referred to as directory files. Whenever a database is created, these directories are updated with the database’s name and alias. If specified, a comment and code set values are also stored in these directories.

7. The database configuration file for the database is initialized.

Some of the parameters in the database configuration file (such as code set, territory, and collating sequence) will be set using values that were specified as input for the `CREATE DATABASE` command or the Create Database Wizard; others are assigned system default values.

8. Four schemas are created.

Schemas are objects that are used to logically classify and group other objects in the database. Once the system catalog tables and views are created, the following schemas are created: `SYSIBM`, `SYSCAT`, `SYSSTAT`, and `SYSFUN`. A special user named `SYSIBM` is made the owner of each.
9. A set of utility programs are bound to the database.
Before some of the DB2 utilities available can work with a database, the packages needed to run those utilities must be created. Such packages are created by binding a set of predefined DB2 Database Manager bind files to the database (the bind files used are stored in the utilities bind list file `db2ubind.lst`).

10. Authorities and privileges are granted to the appropriate users.
To connect to and work with a particular database, a user must have the authorities and privileges needed to use that database. Therefore, unless otherwise specified, whenever a new database is created the following authorities and privileges are granted:

- Database Administrator (DBADM) authority as well as CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, IMPLICIT_SCHEMA, and LOAD privileges are granted to the user who created the database.
- USE privilege on the table space USERSPACE1 is granted to the group PUBLIC.
- CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA privileges are granted to the group PUBLIC.
- SELECT privilege on each system catalog table is granted to the group PUBLIC.
- EXECUTE privilege on all procedures found in the SYSIBM schema is granted to the group PUBLIC.
- EXECUTE WITH GRANT privilege on all functions found in the SYSFUN schema is granted to the group PUBLIC.
- BIND and EXECUTE privileges for each successfully bound utility are granted to the group PUBLIC.

11. Several autonomic features are enabled.
To help make management easy, whenever a new database is created, the following autonomic features are enabled:

- Automatic Maintenance (database back up operations, table and index reorganization, data access optimization, and statistics profiling)
- Self-Tuning Memory Manager (package cache, locking memory, sort memory, database shared memory, and buffer pool memory)
12. The Configuration Advisor is launched.

The Configuration Advisor is a tool designed to help you tune performance and balance memory requirements for a database by suggesting which configuration parameters to modify based on information you provide about the database. In DB2 9.7, the Configuration Advisor is automatically invoked whenever you create a database, unless the default behavior is changed by assigning the value NO to the `DB2_ENABLE_AUTOCONFIG_DEFAULT` registry variable.
A closer look at table spaces

Earlier, we saw that table spaces are used to control where data is physically stored and to provide a layer of indirection between database objects (such as tables, indexes, and views) and one or more containers, such as directories, files, or raw devices, in which the object’s data actually resides. Data is transferred to and from containers in 4 KB, 8 KB, 16 KB, or 32 KB blocks called *pages* and when a table space spans multiple containers, data is written in groups of pages called *extents*, in a round-robin fashion, to each container assigned to that table space. This helps balance data across all containers used. Figure 11 shows the relationship between pages, extents, and table space containers.

![Diagram showing how data is written to table space containers](ICO-IMG-000054)
**Note:** When multiple containers are used to store an SMS table space’s data, the maximum amount of data that each container can hold is determined by the smallest container used. For example, if a table space uses one container that is 10 MB (megabytes) in size and a second container that is 12 MB in size, 2 MB of the second container will not be usable; the maximum amount of storage available to the table space will be 20 MB. Therefore, container sizes should be equal whenever possible.

Two types of table spaces can exist: System Managed Space (SMS) table spaces and Database Managed Space (DMS) table spaces. With SMS table spaces, only directory containers can be used for storage, and the operating system’s file manager is responsible for controlling how that space is used. The SMS storage model consists of many files (each representing a table, index, or long data object) that reside within the file system space — the user decides on the location of the files, the DB2 Database Manager assigns the files their names, and the file system is responsible for managing their growth. With DMS table spaces, only file and/or device containers can be used for storage, and the DB2 Database Manager is responsible for controlling how the space is used. In DB2 9.7, the initial allocation of space for an object in a DMS table space is two extents; the initial allocation of space for an object in an SMS table space is one extent.

Both SMS and DMS table spaces are classified according to the type of data they are intended to store; three classifications exist: regular, large, and temporary. Regular data and index data reside in regular table spaces, whereas long field data and large object data can reside in large table spaces — but only if DMS table spaces are used. (The use of large table spaces is optional given that large data can reside in regular table spaces as well.) Temporary table spaces are further classified as being either system temporary or user temporary — system temporary table spaces are used to store internal temporary data generated when some types of operations are performed (for example, sorting data, reorganizing tables, creating indexes, and joining tables), whereas user temporary table spaces are used to store declared global temporary tables, which in turn are used to store application-specific data for a brief period of time.

If a database is enabled for automatic storage, one other type of table space — an Automatic Storage (AS) table space — can exist. Although at first glance, AS table spaces appear to be a third type of table space, they are really just an extension of SMS and DMS table spaces: regular and large table spaces are created as DMS table spaces with one or more file containers; system and user temporary table
spaces are created as SMS table spaces with one or more directory containers. Unlike when SMS and DMS table spaces are defined, no container definitions are needed for automatic storage table spaces; the DB2 Database Manager assigns containers to automatic storage table spaces automatically.

Some of the more common differences between SMS and DMS/AS table spaces can be seen in Table 1.

<table>
<thead>
<tr>
<th>SMS table spaces</th>
<th>DMS/AS table spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage space is allocated and managed by the operating system's file manager.</td>
<td>Storage space is allocated, if so specified, and managed by the DB2 Database Manager.</td>
</tr>
<tr>
<td>Only directory containers can be used for storage; file and device containers cannot be used.</td>
<td>File or device containers can be used as storage; directory containers cannot be used.</td>
</tr>
<tr>
<td>No additional containers can be added to a table space (using the ALTER TABLESPACE SQL statement) once it has been created.</td>
<td>Additional containers can be added to a table space after it has been created. When new containers are added, existing data can automatically be rebalanced across the new set of containers to retain optimal I/O efficiency.</td>
</tr>
<tr>
<td>Storage space is allocated as it is needed.</td>
<td>Storage space is preallocated.</td>
</tr>
<tr>
<td>A container's size cannot be changed once a table space has been created.</td>
<td>A container's size can be increased or decreased after a table space has been created.</td>
</tr>
<tr>
<td>Regular data and long data are stored in the same table space.</td>
<td>Regular data and long data can be split across multiple table spaces (regular data can reside in one table space while long data resides in another).</td>
</tr>
<tr>
<td>Table spaces are easier to create and manage.</td>
<td>Table access is slightly faster, so overall performance is better.</td>
</tr>
</tbody>
</table>

In the early days of DB2 LUW, DMS table spaces using raw devices as storage containers could deliver better performance than SMS table spaces or DMS table spaces using file containers. As technology has improved the performance gap between DMS table spaces using raw devices as storage containers and DMS table spaces using files as containers has narrowed to the point that IBM now recommends using DMS table spaces that rely on file containers. And starting in version 8.2, these types of table spaces can be configured to grow automatically, as needed.
With DB2 version 8.2 and later, it is also possible to create a DB2 database that uses automatic storage; when automatic storage is used, you simply assign one or more storage locations to the database and the DB2 Database Manager will build its table spaces across the pool of available storage.

**Note:** Ideally, a DB2 LUW database deployed in a SAN environment will use either automatic storage or auto-resizing DMS file table spaces to house user data. Automatic storage, SMS, or fixed size DMS file table spaces should be used to store temporary data.
Creating additional table spaces

It was mentioned earlier that when a DB2 database is created, one buffer pool named IBMDEFAULTBP is created, and three table spaces are created and associated with this buffer pool as part of the database initialization process. These three table spaces are sufficient for small databases; however, large databases are usually composed of many different buffer pool and table space objects. Additional table spaces can be created by executing the `CREATE TABLESPACE` SQL statement or by using the Create Table Space Wizard, which can be activated by selecting the appropriate action from the Table Spaces menu found in the Control Center. Figure 12 shows the Control Center menu items that must be selected to activate the Create Table Space Wizard; Figure 13 on page 59 shows how the first page of the Create Table Space Wizard might look like when it is first activated.

![Figure 12 Invoking the Create Table Space Wizard from the Control Center](ICO-IMG-000063)
Modifying existing table spaces

Because SMS table spaces rely on the operating system for physical storage space management, they rarely need to be modified after they have been successfully created. DMS table spaces, on the other hand, have to be monitored closely to ensure that the fixed-size preallocated file(s) or physical raw device(s) that they use for storage always have enough free space available to meet the database’s needs. When the amount of free storage space available to a DMS table space becomes dangerously low (typically less than 10 percent), you can add more free space either by increasing the size of one or more of its containers or by adding one or more new containers to it. Existing table space containers can be resized, new containers can be made available to an existing table space, and an existing table space’s properties can be changed by executing the `ALTER TABLESPACE` SQL statement.
Table spaces can also be altered using the Alter Table Space dialog box, which can be activated by selecting the appropriate action from the Table Spaces menu found in the Control Center. Figure 14 shows the Control Center menu items that must be selected in order to activate the Alter Table Space dialog box; Figure 15 on page 61 shows how the first page of the Alter Table Space dialog box might look like when it is first activated.

![Figure 14 Invoking the Alter Table Space dialog box from the Control Center](ICO-IMG-000065)
Adding new containers to existing automatic storage table spaces

Earlier, we saw that if a database is enabled for automatic storage, the container- and space-management characteristics of its table spaces are determined by the DB2 Database Manager. And although the ALTER TABLESPACE command can be used to add new containers to existing DMS table spaces, it cannot be used to add new containers to automatic storage stable spaces. So how can you add new storage paths to the collection of paths that are used for automatic storage table spaces once a database has been created? To perform this type of operation, you must use the ALTER DATABASE statement. The basic syntax for this statement is:

```
ALTER DATABASE [DatabaseName]
ADD STORAGE ON '[Container]' , ...
```

where:

- `DatabaseName` — Identifies the database, by name, that is to have new containers added to its pool of containers that are used for automatic storage.
Introduction to DB2 for Linux, UNIX, and Windows

♦  **Container** — Identifies one or more new storage locations (containers) that are to be added to the collection of storage locations that are used for automatic storage table spaces.

Thus, if you wanted to add the storage locations `/data/path1` and `/data/path2` to a database named `SAMPLE` that is configured for automatic storage and resides on a UNIX system, you could do so by executing an `ALTER DATABASE` SQL statement that looks like this:

```
ALTER DATABASE sample
ADD STORAGE '/data/path1', '/data/path2'
```
A closer look at transaction logging

A transaction (also known as a unit of work) is a sequence of one or more SQL operations grouped together as a single unit, usually within an application process. The initiation and termination of a single transaction define points of data consistency within a database; either the effects of all operations performed within a transaction are applied to the database and made permanent (committed), or the effects of all operations performed are backed out (rolled back), and the database is returned to the state it was in before the transaction was initiated.

In most cases, transactions are initiated the first time an executable SQL statement is executed after a connection to a database has been made or immediately after a pre-existing transaction has been terminated. Once initiated, transactions can be implicitly terminated using a feature known as “automatic commit” (in which case, each executable SQL statement is treated as a single transaction, and any changes made by that statement are applied to the database if the statement executes successfully or are discarded if the statement fails), or they can be explicitly terminated by executing the COMMIT or the ROLLBACK SQL statement.

Transaction logging is a process that is used to keep track of changes made to a database by transactions, as they occur. Each time an update or a delete operation is performed, the page containing the record to be updated/deleted is retrieved from storage and copied to the appropriate buffer pool, where it is then modified by the update/delete operation. (If a new record is created by an insert operation, that record is created directly in the appropriate buffer pool.) Once the record has been modified (or inserted), a record reflecting the modification/insertion is written to the log buffer, which is simply a designated storage area in memory. (The actual amount of memory reserved for the log buffer is controlled by the logbufsiz database configuration parameter.) If an insert operation is performed, a record containing the new row is written to the log buffer; if a delete operation is performed, a record containing the row’s original values is written to the log buffer; and if an update operation is performed, a record containing the row’s original values, combined with the row’s new values, is written to the log buffer. (If replication has not been enabled, an Exclusive OR operation is performed using the “before” and “after” rows and the results are written to the log buffer.) These kinds of records, along with records
that indicate whether the transactions that were responsible for making the changes were committed or rolled back, make up the majority of the records stored in the log buffer.

Whenever buffer pool I/O page cleaners are activated (I/O page cleaners are special agents that are responsible for writing pages to disk at predetermined intervals), the log buffer becomes full, or a transaction is terminated (by being committed or rolled back), all records stored in the log buffer are immediately written to one or more log files stored on disk. This is done to minimize the number of log records that might get lost in the event a system failure occurs. As soon as all log records associated with a particular transaction have been externalized to one or more log files, the effects of the transaction itself are recorded in the database, such as executed against the appropriate table space containers for permanent storage. The modified data pages remain in memory, where they can be quickly accessed if necessary — eventually, they will be overwritten as newer pages are retrieved from storage. The transaction logging process can be seen in Figure 16 on page 65.
Because multiple transactions may be working with a database at any given point in time, a single log file may contain log records that belong to several different transactions. Therefore, to keep track of which log records belong to which transactions, every log record is assigned a special “transaction identifier” that ties it to the transaction that created it. By using transaction IDs, log records associated with a particular transaction can be written to one or more log files at any time, without impacting data consistency — eventually, the execution of the COMMIT or ROLLBACK statement that terminates the transaction will be logged as well.

Because log records are externalized frequently and because changes made by a particular transaction are only externalized to the database when the transaction itself is successfully terminated, the ability to return a database to a consistent state after a failure occurs is guaranteed — when the database is restarted, log records are
analyzed, and each record that has a corresponding COMMIT record is reapplied to the database; every record that does not have a corresponding COMMIT record is either ignored or backed out (which is why “before” and “after” information is recorded for all update operations).

Logging strategies

When a database is first created, three log files, known as primary log files, are allocated as part of the creation process. On Linux and UNIX platforms, these log files are 1,000 4K (kilobyte) pages in size; on Windows platforms, these log files are 250 4K pages in size. However, the number of primary log files used, along with the amount of data each is capable of holding, is controlled by the logprimary and logfilsiz parameters in the database’s configuration file. The way in which all primary log files created are used is determined by the logging strategy chosen for the database. Two very different strategies, known as circular logging and archival logging, are available.

Circular logging

When circular logging is used, records stored in the log buffer are written to primary log files in a circular sequence. Log records are written to the current “active” log file, and when that log file becomes full, it is marked as “unavailable.” At that point, DB2 makes the next log file in the sequence the active log file and begins writing log records to it; when that log file becomes full, the process is repeated. In the meantime, as transactions are terminated and their effects are externalized to the database, their corresponding log records are released because they are no longer needed. When all records stored in an individual log file are released, that file is marked as being “reusable,” and the next time it becomes the active log file, its contents are overwritten with new log records.

Although primary log files are not marked reusable in any particular order (they are marked reusable when they are no longer needed), they must be written to in sequence. So what happens when the logging process cycles back to a primary log file that was marked as being “unavailable”? When this occurs, DB2 will allocate what is known as a secondary log file and begin writing log records to it. As soon as the secondary log file becomes full, DB2 will poll the primary log file again, and if its status is still “unavailable,” another secondary log file is allocated and filled. This process will continue until either the desired primary log file becomes “reusable” or the number of
secondary log files created matches the number of secondary log files allowed (designated by the `logsecond` database configuration parameter). If the former occurs, DB2 will begin writing log records to the appropriate primary log file, and logging will pick up where it left off in the logging sequence. In the meantime, log records stored in the secondary log files are eventually released, and when all connections to the database have been terminated, any secondary log files that were created are destroyed. On the other hand, if the maximum number of secondary log files allowed has been allocated, and the desired primary log file is still unavailable, all database activity will stop, and the following message will be generated:

**SQL0964C The transaction log for the database is full.**

With the default configuration, up to two secondary log files will be created if necessary and their size will be the same as that of each primary log file used. Circular logging is illustrated in Figure 17.

---

**Figure 17  Circular logging**

By default, when a new database is created, circular logging is the logging strategy used.
Archival logging

Like circular logging, when archival logging (also known as log retention logging) is used, log records stored in the log buffer are written to the primary log files that have been preallocated. However, unlike with circular logging, these log files are never reused. Instead, when all records stored in an individual log file are released, that file is marked as being “archived” rather than as being “reusable,” and the only time it is used again is if it is needed for a roll-forward recovery operation. As soon as an active log file becomes full, DB2 allocates a new log file, in sequence, and that file can be used as a primary or a secondary log file, depending on what is needed when that sequence number is hit. This process continues as long as there is sufficient disk space available.

Because any number of primary log files can exist when archival logging is used, they are classified according to their current state and storage location. Log files containing records associated with transactions that have not yet been committed or rolled back are known as active log files and reside in the active log directory (or device). Log files containing records associated with completed transactions (i.e., transactions that have been externalized to the database) that reside in the active log directory are known as online archive log files. Log files containing records that are associated with completed transactions that have been moved to a storage location other than the active log directory are known as offline archive log files. Offline archive files can be moved to their storage location automatically by assigning the appropriate value (USEREXIT, DISK, TSM, or VENDOR) to the logarchmeth1 or logarchmeth2 database configuration parameter. (In this case, DB2 will attempt to move log files to the archive location specified as soon as they become full.) Archival logging is illustrated in Figure 18 on page 69.
A closer look at transaction logging

Introduction to DB2 for Linux, UNIX, and Windows

It is important to note that if both the logarcmeth1 and the logarcmeth2 database configuration parameters are assigned values, DB2 will store copies of offline archive log files in two locations; both copies will be kept in sync.

Other logging considerations

Along with specifying the logging strategy to employ, several database configuration parameters can be used to control a database’s logging behavior. The following items should be taken into consideration when configuring a database for transaction log management.

Infinite logging

You would think that you could avoid running out of log space simply by configuring a database to use a large number of secondary log files, if needed. However, the maximum number of primary and secondary log files allowed (logprimary + logsecond) is 256, and if the size of your log files is relatively small, you can still run out of log space quickly when transaction workloads become heavy. Furthermore, you want to avoid allocating a large number of secondary log files if possible because performance is impacted each time a log file has to be allocated. Ideally, you want to allocate enough primary log files to handle most situations, and you want to use just enough secondary log files to handle peaks in transaction workloads.
If you are concerned about running out of log space, and you want to avoid allocating a large number of secondary log files, you can configure a database to use what is known as infinite logging. To enable infinite logging for a database, you simply set the `logsecond` database configuration parameter to -1.

In order to use infinite logging, a database must be configured to use archival logging. This means that if a problem occurs and a long running transaction needs to be rolled back, DB2 may have to access one or more log files that have already been archived, which, in turn, will negatively impact performance. And if the database needs to be restarted and some of the log files needed have been archived, the recovery process may take longer. To control the impact infinite logging can have on rollback and recovery operations, you can limit the number of logs, or total log space that an active transaction can use by assigning appropriate values to the `max_log` and `num_log_span` database configuration parameters.

Log mirroring

With DB2 version 8.1 and later, you have the ability to configure a database such that DB2 will simultaneously create and update active log files in two different locations. If you store active log files in one location and mirror them in another, separate location, database activity can continue if a disk failure or human error causes log files in one location to become inaccessible. (Mirroring log files may also aid in database recovery.) To enable log file mirroring, you simply assign the fully qualified name of the mirror log location (path) to the `mirrorlogpath` database configuration parameter. Ideally, the mirror log path used should refer to a physical location (disk) that does not see a large amount of disk I/O and that is separate from the physical location used to store primary log files.

If an error is encountered during attempts to write to either the active log path or the mirror log path, DB2 will mark the failing path as “bad,” write a message to the administration notification log, and write subsequent log records to the remaining “good” log path only. When DB2 allocates storage for its next primary log file, it will make a second attempt to write to both log paths. If successful, dual logging will continue. If not, DB2 will not attempt to use the “bad” path again until the next log file is accessed for the first time. There is no attempt to synchronize the log paths, but DB2 keeps track of each access error that occurs, so that the correct paths will be used when log files are archived. If a failure occurs while writing to the remaining “good” path, the database shuts down.
Archival logging failover

Earlier, we saw that a database can be configured to use archival logging and that archived log files can be automatically moved from the active log directory to another location when they become full — provided the value USEREXIT, DISK, TSM, or VENDOR is assigned to the logarcmeth1 or the logarcmeth2 database configuration parameter. If a database has been configured to take advantage of this functionality and for some reason (for example, because of a media problem), the archived log files cannot be moved to either the primary or the secondary (if set) archive destination, they will remain in the active log directory until the problem is resolved. This behavior, in turn, can cause a database to run out of log space if the active log directory is not large enough to accommodate the archived log files being generated.

To prevent such a problem from occurring, DB2 can be configured to store archived log files in an alternate location if, for some reason, the primary location becomes unavailable or the log archival method chosen fails. To take advantage of this functionality, you simply assign the fully qualified name of an alternate storage location (path) to the failarchpath database configuration parameter. The location specified will act as a temporary storage area for the log files until the primary location or log archival method that failed becomes available again; at that time, any log files stored in the storage area will automatically be moved to the primary location.

Controlling how “disk full” errors are handled

When archival logging is used and archived log files are not moved from the active log directory to another location, the disk where the active log directory resides can quickly become full. By default, when this happens, transactions will receive a disk full error and be rolled back. But what if, instead of the current transaction being terminated, you were given the chance to manually move or delete files to make more room available? That's the purpose behind the blk_log_dsk_ful database configuration parameter; if this parameter is set to YES, applications will hang if the DB2 Database Manager receives a disk full error when it attempts to create a new log file in the active log directory. The DB2 Database Manager will then attempt to create the log file every five minutes until it succeeds—after each attempt, a message is written to the Administration Notification Log. (The only way that you can confirm that an application is hung because of a disk full condition is to monitor this log.) Until the log file is successfully created, applications attempting to insert or update data
will not be permitted to commit their transactions. Read-only queries may not be directly affected; however, if a query needs to access data that is locked by an update request or a data page that is fixed in the buffer pool by the updating application, read-only queries will also appear to hang.

To resolve a disk full situation, you simply move old log files to another location or enlarge the current file system. As soon as the needed space becomes available, new log files can be created, and all hung applications will be able to continue processing.
Configuring a DB2 LUW database environment

Earlier, we saw that a DB2 database environment contains a set of system objects that are used to control how resources are allocated for a server, an instance, or a database. Along with the comprehensive set of registry variables, DB2 uses an extensive array of configuration parameters to control how system resources are allocated and utilized on behalf of an instance and a database. However, the default values provided for many of the registry variables and configuration parameters that make up these system objects were produced with very simple systems in mind. (The goal was for DB2 to run out of the box, on virtually any platform, not for DB2 to run optimally on the platform on which it is installed.) Thus, while the default values provided are sufficient to meet most needs, you can often improve overall system and application performance simply by changing the values of one or more registry variables and/or configuration parameters.

Configuring servers

During normal operation, the behavior of the DB2 Database Manager is controlled, in part, by a collection of values that define the DB2 operating environment. Some of these values are operating system environment variables, and others are special DB2-specific system-level values known as environment or registry variables. Registry variables provide a way to centrally control the database environment. Three different registry profiles are available, and each controls the database environment at a different level. The registry profiles available are as follows:

- **The DB2 Global — Level Profile Registry.** All machine-wide environment variable settings are kept in this registry; one global-level profile registry exists on each DB2 workstation. If an environment variable is to be set for all instances, this profile registry is used.

- **The DB2 Instance — Level Profile Registry.** The environment variable settings for a particular instance are kept in this registry; this is where the majority of the DB2 environment variables are set. (Values defined in this profile registry override any corresponding settings in the global-level profile registry.)
The DB2 Instance Node — Level Profile Registry. This profile registry level contains variable settings that are specific to a partition (node) in a multipartitioned database environment. (Values defined in this profile registry override any corresponding settings in the global-level and instance-level profile registries.)

Note: DB2 looks for environment variable values in the DB2 global-level profile registry first, then in the DB2 instance-level profile registry, and finally, in the DB2 instance node–level profile registry. (Additional values may be set in individual sessions, in which case DB2 will see these values last.)

A wide variety of registry variables are available, and they vary depending on the operating system being used. A complete listing can be found in Appendix B of the IBM DB2 Troubleshooting and Tuning Database Performance product documentation.

So how do you determine which registry variables have been set, and what they have been set to? Or more importantly, how do you assign values to one or more registry variables? One way is by executing the **db2set** system command. The basic syntax for this command is:

```
db2set
  <[Variable] = [Value]>
  <-g | -gl | -i [InstanceName]>
  <-all>
  <-null>
  <-r [InstanceName]>
  <-l | -lr>
  <-v>
  <-ul | -ur>
  <-h | -?>
```

where:

- **Variable** — Identifies the registry variable whose value is to be displayed, set, or removed.
- **Value** — Identifies the value that is to be assigned to the registry variable specified. If no value is provided, but a registry variable is specified, the registry variable specified is deleted.
- **InstanceName** — Identifies the instance profile with which the specified registry variable is associated.
● **DASNode** — Identifies the name of the node where the DB2 Administration Server instance resides.

● **UserID** — Identifies the authentication ID that will be used to attach to the DB2 Administration Server instance.

● **Password** — Identifies the password (for the authentication ID) that will be used to attach to the DB2 Administration Server instance.

All other options shown with this command are described in Table 2.

**Table 2  db2set command options (page 1 of 2)**

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g</td>
<td>Indicates that a global profile variable is to be displayed, set, or removed.</td>
</tr>
<tr>
<td>-gl</td>
<td>Indicates that a global profile variable stored in LDAP is to be displayed, set, or removed. This option is only effective if the registry variable DB2_ENABLE_LDAP has been set to YES.</td>
</tr>
<tr>
<td>-i</td>
<td>Indicates that an instance profile variable is to be displayed, set, or removed.</td>
</tr>
</tbody>
</table>
| -all   | Indicates that all occurrences of the registry variable, as defined in the following, are to be displayed:  
  - The environment (denoted by [-e])  
  - The node-level registry (denoted by [-n])  
  - The instance-level registry (denoted by [-i])  
  - The global-level registry (denoted by [-g]) |
| -null  | Indicates that the value of the variable at the specified registry level is to be set to NULL. |
| -r     | Indicates that the profile registry for the given instance is to be reset. |
| -n     | Indicates that a remote DB2 Administration Server instance node name is specified. |
| -u     | Indicates that an authentication ID that will be used to attach to the DB2 Administration Server instance is specified. |
| -p     | Indicates that a password for the authentication ID specified is provided. |
| -l     | Indicates that all instance profiles will be listed. |
| -lr    | Indicates that all registry variables supported will be listed. |
| -v     | Indicates that the db2set command is to be executed in verbose mode. |
It is important to note that if the `db2set` command is executed without options, a list containing every registry variable that has been set for the current (default) instance, along with its value, will be returned.

Thus, if you wanted to find out which registry variables have been set for each profile available, you could do so by executing a `db2set` command that looks like this:

```
db2set -all
```

On the other hand, if you wanted to see the current value of the `DB2_PARALLEL_IO` registry variable for all DB2 instances, you could do so by executing a `db2set` command that looks something like this:

```
db2set -l DB2_PARALLEL_IO
```

And finally, if you wanted to assign the value 7 to the `DB2_PARALLEL_IO` registry variable for all DB2 instances on a server, you could do so by executing a `db2set` command that looks something like this:

```
db2set -g DB2_PARALLEL_IO=7
```

Another way to view and/or change registry variable settings is by using a tool known as the DB2 Registry management tool. The DB2 Registry management tool is activated by selecting the DB2 Registry action from the Configure menu found in the Configuration Assistant. Figure 19 on page 77 shows the Configuration Assistant menu items that must be selected in order to activate the DB2 Registry management tool. Figure 20 on page 78 shows how the main dialog box of the DB2 Registry management tool might look after it has been activated.
Figure 19  Invoking the DB2 Registry management tool from the Configuration Assistant


### Figure 20  DB2 Registry management tool dialog box

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global</strong></td>
<td></td>
</tr>
<tr>
<td>DB2_EXTRA_SECURITY</td>
<td>YES</td>
</tr>
<tr>
<td>DB2.ADMIN_SERVER</td>
<td>DB20A300</td>
</tr>
<tr>
<td>DB2INSTDEF</td>
<td>DB2</td>
</tr>
<tr>
<td>DB2PATH</td>
<td>C:\DB2\ESE\GOLLIB</td>
</tr>
<tr>
<td>DB2SYSTEM</td>
<td>DB_SERVER</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td></td>
</tr>
<tr>
<td>DB2_ENABLE_AUTOCONFIG_DEFAULT</td>
<td>NO</td>
</tr>
<tr>
<td>DB2_XML_RUNSTATS_PATHID_K</td>
<td>300</td>
</tr>
<tr>
<td>DB2_XML_RUNSTATS_PATHVALUE_K</td>
<td>300</td>
</tr>
<tr>
<td>DB2ACCOUNTNAME</td>
<td>db_server/db2admin</td>
</tr>
<tr>
<td>DB2COMM</td>
<td>TCP/IP, IPPE</td>
</tr>
<tr>
<td>DB2INSTOWNER</td>
<td>db_server</td>
</tr>
<tr>
<td>DB2INSTPROF</td>
<td>C:\DB2\ESE\GOLLIB</td>
</tr>
<tr>
<td>DB2PORTRANGE</td>
<td>50000-80000</td>
</tr>
</tbody>
</table>

13 of 13 items displayed  View

ICO-IMG-000058
Configuring instances

Whenever an instance is created, a corresponding DB2 Database Manager configuration file is also created and initialized as part of the instance creation process. Each DB2 Database Manager configuration file is made up of approximately 85 different parameter values, and most control the amount of system resources that are allocated to a single DB2 Database Manager instance.

You can view the contents of the DB2 Database Manager configuration file for the current instance by executing the GET DATABASE MANAGER CONFIGURATION command. The syntax for this command is:

```
GET [DATABASE MANAGER | DB MANAGER | DBM]
[CONFIGURATION | CONFIG | CFG]
<SHOW DETAIL>
```

Thus, if you wanted to view the contents of the DB2 Database Manager configuration file for the current instance, you could do so by executing a GET DATABASE MANAGER CONFIGURATION command that looks like this:

```
GET DBM CFG
```

You can change the value assigned to a particular DB2 Database Manager configuration file parameter for the current instance by executing the UPDATE DATABASE MANAGER CONFIGURATION command. The syntax for this command is:

```
UPDATE [DATABASE MANAGER | DB MANAGER | DBM]
[CONFIGURATION | CONFIG | CFG]
USING [[Parameter] [Value] | [Parameter] [Value] AUTOMATIC | [Parameter] AUTOMATIC | [Parameter] MANUAL,...]
<IMMEDIATE | DEFERRED>
```

where:

- **Parameter** — Identifies one or more DB2 Database Manager configuration parameters (by keyword) whose values are to be modified. (In many cases, the keyword for a parameter is the same as the parameter name itself.)
- **Value** — Identifies the new value or values that are to be assigned to the DB2 Database Manager configuration parameter(s) specified.
If the AUTOMATIC keyword is specified as the value for a particular parameter, DB2 will automatically adjust the parameter value to reflect the current resource requirements. If a value is specified along with the AUTOMATIC keyword, the value provided may influence the automatic calculations performed.

If the DEFERRED clause is specified with the UPDATE DATABASE MANAGER CONFIGURATION command, changes made to the DB2 Database Manager configuration file will not take effect until the instance is stopped and restarted. If the IMMEDIATE clause is specified instead, or if neither clause is specified, all changes made to the DB2 Database Manager configuration file will take effect immediately — provided that the necessary resources required are available.

So if you wanted to configure the current instance such that the maximum number of applications that can be executing concurrently at any given point in time is 100, you could do so by executing an UPDATE DATABASE MANAGER CONFIGURATION command that looks like this:

```
UPDATE DBM CFG USING MAXCAGENTS 100
```

The contents of a DB2 Database Manager configuration file can also be viewed or altered using the DBM Configuration dialog box, which can be activated by selecting the Configure Parameters action from the Instances menu found in the Control Center. Figure 21 on page 81 shows the Control Center menu items that must be selected to activate the DBM Configuration dialog box. Figure 22 on page 82 shows how this dialog box might look after it has been activated.
Figure 21  Invoking the DBM Configuration dialog box from the Control Center
Configuring databases

Just as a DB2 Database Manager configuration file is created and initialized whenever a new instance is created, a database configuration file is created and initialized each time a new database is created. Each database configuration file is made up of several different parameters, and just as most DB2 Database Manager instance configuration parameters control the amount of system resources that will be allocated to a single DB2 Database Manager...
instance, many of the database configuration file parameters control the amount of system resources that will be allocated to a database during normal operation.

The contents of the database configuration file for a particular database can be displayed by executing the `GET DATABASE CONFIGURATION` command. The syntax for this command is:

```
GET [DATABASE | DB] [CONFIGURATION | CONFIG | CFG]
FOR [DatabaseAlias]
<SHOW DETAIL>
```

where:

- `DatabaseAlias` — Identifies the alias assigned to the database that configuration information is to be displayed for.

Thus, if you wanted to view the contents of the database configuration file for a database named `SAMPLE`, you could do so by executing a `GET DATABASE CONFIGURATION` command that looks like this:

```
GET DB CFG FOR sample
```

The value assigned to a particular database configuration file parameter can be changed by executing the `UPDATE DATABASE CONFIGURATION` command. The syntax for this command is:

```
UPDATE [DATABASE | DB]
[CONFIGURATION | CONFIG | CFG]
FOR [DatabaseAlias]
USING [[Parameter] [Value] | [Parameter] [Value] AUTOMATIC | [Parameter] AUTOMATIC | [Parameter] MANUAL ,...]
<IMMEDIATE | DEFERRED>
```

where:

- `DatabaseAlias` — Identifies the alias assigned to the database for which configuration information is to be modified.
- `Parameter` — Identifies one or more database configuration parameters (by keyword) whose values are to be modified. (In many cases, the keyword for a parameter is the same as the parameter name itself.)
- `Value` — Identifies the new value(s) that are to be assigned to the database configuration parameter(s) specified.
Once again, if the **AUTOMATIC** keyword is specified as the value for a particular parameter, DB2 will automatically adjust the parameter value to reflect the current resource requirements. If a value is specified along with the **AUTOMATIC** keyword, the value provided may influence the automatic calculations performed.

If the **DEFERRED** clause is specified with the **UPDATE DATABASE CONFIGURATION** command, changes made to the database configuration file will not take effect until all connections to the corresponding database have been terminated and a new connection is established. If the **IMMEDIATE** clause is specified instead, or if neither clause is specified, all changes made to the database configuration file will take effect immediately — provided the necessary resources are available. (Applications running against a database at the time database configuration changes are made will see the change the next time an SQL statement is executed.)

So if you wanted to configure a database named **SAMPLE** such that any application connected to the database will wait up to 100 seconds to acquire a lock before rolling back the current transaction, you could do so by executing an **UPDATE DATABASE CONFIGURATION** command that looks like this:

```sql
UPDATE DB CFG FOR sample USING LOCKTIMEOUT 100
```

You can view or alter the contents of a database configuration file by using the Database Configuration dialog box, which can be activated by selecting the Configure Parameters action from the Databases menu found in the Control Center. Figure 23 on page 85 shows the Control Center menu items that must be selected to activate the Database Configuration dialog box. Figure 24 on page 86 shows how this dialog box might look after it has been activated.
Figure 23  Invoking the Database Configuration dialog box from the Control Center
Figure 24  Database Configuration dialog box
This chapter introduces the EMC foundation products, some of which are discussed in this TechBook, that work in combined Symmetrix and open systems environments:

- Introduction ........................................................................................ 88
- Symmetrix hardware and EMC Enginuity features ......................... 90
- EMC Solutions Enabler base management ..................................... 95
- EMC Change Tracker ...................................................................... 98
- EMC Symmetrix Remote Data Facility ........................................... 99
- EMC TimeFinder ............................................................................. 117
- EMC Replication Manager ......................................................... 123
- EMC Storage Resource Management ............................................ 126
- EMC PowerPath ........................................................................... 130
- EMC Open Replicator ................................................................. 132
- EMC Virtual Provisioning ............................................................ 133
- EMC Fully Automated Storage Tiering (FAST) ............................ 137
EMC Foundation Products

Introduction

EMC provides many hardware and software products that support application environments on Symmetrix® storage systems. The following products, which are highlighted and discussed, have been used and/or tested with DB2 for Linux, UNIX, and Windows databases.

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 EMC Symmetrix VMAX™ Series with Enginuity™, the Symmetrix DMX™ Direct Matrix Architecture® systems and earlier Symmetrix system array models. The EMC Symmetrix array is a fully redundant, high-availability storage processor, providing nondisruptive component replacements and code upgrades. Symmetrix arrays features high levels of performance, data integrity, reliability, and availability.

EMC Enginuity Operating Environment — Enginuity enables interoperation between the latest Symmetrix platforms and previous generations of Symmetrix systems 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, ranging from HBAs and drivers to switches and tape systems.

EMC Solutions Enabler — Solutions Enabler is a package that contains the SYMAPI runtime libraries and the SYMCLI command line interface. SYMAPI provides the interface to the EMC Enginuity operating environment; SYMCLI is a set of commands that can be invoked from the command line or within scripts. These commands can be used to monitor device configuration and status and to perform control operations on devices and data objects within a storage complex.

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.

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...
EMC Foundation Products

system. The SRDF component extends the basic SYMCLI command set of Solutions Enabler to include commands that specifically manage SRDF.

**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 enable LUN-based 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.

**Solutions Enabler Storage Resource Management (SRM) component** — The SRM component extends the basic SYMCLI command set of Solutions Enabler to include commands that allow users to systematically find and examine attributes of various objects on the host, within a specified relational database, or in the EMC enterprise storage. The SRM commands provide mapping support for relational databases, file systems, logical volumes and volume groups, as well as performance statistics.

**EMC PowerPath**® — PowerPath is host-based software that provides I/O path management. PowerPath operates with several storage systems, on several enterprise operating systems and provides failover and load balancing transparent to the host application and database.

**Symmetrix management software** — Enginuity and array-based applications like SRDF and TimeFinder reside in the Symmetrix array and can be managed by host applications. EMC Solutions Enabler includes a CLI interface for SRDF, TimeFinder, device configuration, device mapping, and device masking. Symmetrix Management Console (SMC) provides a browser-based GUI interface on top of Solutions Enabler. EMC Ionix™ ControlCenter® Symmetrix Manager is a central feature of the EMC ControlCenter family of products, which provides a unified view for multiple arrays using a single-pane-of-glass interface. Symmetrix Manager is used to discover, monitor, and configure Symmetrix storage from a single console including the ability to automate key system management, and replication tasks.
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
- Solutions Enabler
- Symmetrix application program interface (API) for mainframe
- Symmetrix-based applications
- Host-based Symmetrix applications
- Independent software vendor (ISV) applications

All Symmetrix systems provide advanced data replication capabilities, full mainframe and open systems support, and flexible connectivity options, including Fibre Channel, FICON, ESCON, Gigabit Ethernet, and iSCSI.

Interoperability between Symmetrix storage systems enables customers to migrate storage solutions from one generation to the next, protecting their investment even as their storage demands expand.

Symmetrix enhanced cache director technology allows configurations of up to 512 GB of cache. The cache can be logically divided into 32 independent regions providing up to 32 concurrent 500 MB/s transaction throughput.

The Symmetrix on-board data integrity features include:

- Continuous cache and on-disk data integrity checking and error detection/correction
- Fault isolation
- Nondisruptive hardware and software upgrades
- Automatic diagnostics and phone-home capabilities

At the software level, advanced integrity features ensure information is always protected and available. By choosing a mix of RAID 1 (mirroring), RAID 1/0, high performance RAID 5 (3+1 and 7+1)
protection and RAID 6, users have the flexibility to choose the protection level most appropriate to the value and performance requirements of their information.

From the perspective of the host operating system, a Symmetrix system appears to be multiple physical devices connected through one or more I/O controllers. The host operating system addresses each of these devices using a physical device name. Each physical device includes attributes, vendor ID, product ID, revision level, and serial ID. The host physical device maps to a Symmetrix device. In turn, the Symmetrix device is a virtual representation of a portion of the physical disk called a hypervolume.

**Symmetrix VMAX platform**

The EMC Symmetrix VMAX Series with Enginuity is the newest addition to the Symmetrix family, and the first high-end system purpose-built for the virtual data center. Based on the Virtual Matrix Architecture™, the Symmetrix VMAX system scales performance and capacity to unprecedented levels, delivers nondisruptive operations, and greatly simplifies and automates the management and protection of information. Advanced tiering via Enterprise Flash, Fibre Channel, and SATA drives allows users to ensure that the right data is on the right storage tier at the right cost.

At the heart of the Symmetrix VMAX system is the Virtual Matrix Architecture, designed to break through the physical boundaries of fixed backplane storage architectures — in a system that can scale to dozens of PBs, support thousands of virtual servers, deliver millions of IOPs, and provide 24x7xforever availability.

The advantages of this unique scale-out architecture, along with new Enginuity operating environment capabilities, are critical for customers transitioning to more of a virtual data center infrastructure. The ability to dynamically scale, while dramatically simplifying and automating operational tasks, is critical to addressing the infrastructure requirements and driving down cost in both virtual and physical deployments.

**Design overview**

The Symmetrix VMAX system design is based on a highly available VMAX system engine with redundant CPU, memory, and connectivity on two directors for fault tolerance. Symmetrix VMAX system engines connect to and scale-out linearly through the Virtual
Matrix Architecture, which allows resources to be shared within and across Symmetrix VMAX system engines. To meet growth requirements, additional engines can be added nondisruptively for efficient and dynamic scaling of capacity and performance that is available to any application on demand. Figure 25 illustrates the architecture and interconnection of the major components in the Symmetrix VMAX storage system.

The Symmetrix VMAX system is the only high-end platform with multi-core processors providing maximum performance and energy-efficient capabilities in each Symmetrix VMAX system engine. This unique feature allows the single engine Symmetrix VMAX system configurations to deliver significantly more performance in a smaller footprint than any other storage array.

**Figure 25**  EMC Symmetrix VMAX Series with Enginuity

Each Symmetrix VMAX system engine (Figure 25, right example) contains two directors with extensive CPU processing power, global cache memory, and a Virtual Matrix Interface for inter-director communications.
The Symmetrix VMAX system engines are configurable to provide maximal and flexible host connectivity and back-end physical drive loops. Front-end port configurations are Fibre Channel, iSCSI and FICON for host connections and Fibre Channel and Gigabit Ethernet for remote replication. Speeds auto-negotiate between 1 and 4 Gigabit per second based on the connection types.

The processing power is provided by dual quad-core 2.33 GHZ Xeon processors from Intel. Each director includes up to 64 GB of memory using eight Cache Memory Modules. Current memory module sizes are 2 GB, 4 GB, and 8 GB, which provide a total capacity of 16, 32, and 64 GB per director or a maximum of 128 GB of physical memory per Symmetrix VMAX system.

To enable multiple director boards and instances to work together as a single system, a high-bandwidth, low latency, nonblocking communication matrix is used. The Symmetrix VMAX Virtual Matrix Interconnect is implemented using the industry-standard Rapid IO (RIO) protocol through two redundant switching elements. On the physical director board, two separate sets of eight lanes of PCI-Express are converted to RIO by the Virtual Matrix Interface. The Matrix Interface Board Enclosure (MIBE) contains two independent matrix switches that provide point-to-point communications between directors. This redundant matrix is used for mirrored writes across directors and for other inter-director signaling and communications.

The Symmetrix VMAX Series provides a distributed architecture that provides near infinite scalability while maintaining a single system to manage. Through the use of the high-speed interconnect, the Symmetrix VMAX system provides the building blocks for EMC high-performance storage systems. This has transformed enterprise storage and is the baseline for how current and future storage systems will be measured.

While the benefits of moving to a distributed architecture are numerous and well understood, it is also a fact that distributed system are typically very difficult to manage. The glue that keeps the distributed architecture of the Symmetrix VMAX system operating as a single system is the Enginuity operating environment.

Many of the new features provided by the new EMC Symmetrix VMAX platform can reduce operational costs for customers deploying DB2 for Linux, UNIX, and Windows environments, as well as enhance functionality to enable greater benefits.
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.
- 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.

The Enginuity operating environment for Symmetrix version 5874 is a feature-rich Enginuity release supporting Symmetrix VMAX storage arrays. With the release of Enginuity 5874, Symmetrix VMAX systems deliver new software capabilities that improve capacity utilization, ease of use, business continuity and security.
EMC Solutions Enabler base management

The EMC Solutions Enabler kit contains all the base management software necessary to provide a host with SYMAPI-shared libraries and the basic Symmetrix command line interface (SYMCLI). Other optional subcomponents in the Solutions Enabler (SYMCLI) series enable users to extend functionality of their Symmetrix systems. Three principle sub-components are:

- Solutions Enabler SYMCLI SRDF, SRDF/CG, and SRDF/A
- Solutions Enabler SYMCLI TimeFinder/Mirror, TimeFinder/CG, TimeFinder/Snap, TimeFinder/Clone
- Solutions Enabler SYMCLI Storage Resource Management (SRM)

These components are discussed later in this chapter.

SYMCLI resides on a host system and is used to perform control operations on devices and data objects on Symmetrix storage arrays. SYMCLI commands are invoked from the host operating system command line or via scripts; when executed, SYMCLI commands invoke low-level channel commands to specialized devices on the Symmetrix called gatekeepers. Gatekeepers are very small devices carved from disks in the Symmetrix that act as SCSI targets for the SYMCLI commands.

SYMCLI commands are also used to monitor device configuration and the status of devices that make up the storage environment. To reduce the number of inquiries from the host to a Symmetrix system, configuration and status information is maintained in a host database file.

Table 3 lists some of the more common SYMCLI base commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>symdg</td>
<td></td>
<td>Performs operations on a device group (dg)</td>
</tr>
<tr>
<td>create</td>
<td></td>
<td>Creates an empty device group</td>
</tr>
<tr>
<td>delete</td>
<td></td>
<td>Deletes a device group</td>
</tr>
<tr>
<td>rename</td>
<td></td>
<td>Renames a device group</td>
</tr>
<tr>
<td>release</td>
<td></td>
<td>Releases a device external lock associated with all devices in a device group</td>
</tr>
<tr>
<td>Command</td>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>Displays a list of all device groups known to this host</td>
</tr>
<tr>
<td>show</td>
<td></td>
<td>Shows detailed information about a device group and any gatekeeper or BCV devices associated with the device group</td>
</tr>
<tr>
<td>symcg</td>
<td></td>
<td>Performs operations on a composite group (cg)</td>
</tr>
<tr>
<td>create</td>
<td></td>
<td>Creates an empty composite group</td>
</tr>
<tr>
<td>add</td>
<td></td>
<td>Adds a device to a composite group</td>
</tr>
<tr>
<td>remove</td>
<td></td>
<td>Removes a device from a composite group</td>
</tr>
<tr>
<td>delete</td>
<td></td>
<td>Deletes a composite group</td>
</tr>
<tr>
<td>rename</td>
<td></td>
<td>Renames a composite group</td>
</tr>
<tr>
<td>release</td>
<td></td>
<td>Releases a device external lock associated with all devices in a composite group</td>
</tr>
<tr>
<td>hold</td>
<td></td>
<td>Hold devices in a composite group</td>
</tr>
<tr>
<td>unhold</td>
<td></td>
<td>Unhold devices in a composite group</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>Displays a list of all composite groups known to this host</td>
</tr>
<tr>
<td>show</td>
<td></td>
<td>Shows detailed information about a composite group, and any gatekeeper or BCV devices associated with the group</td>
</tr>
<tr>
<td>symld</td>
<td></td>
<td>Performs operations on a device in a device group</td>
</tr>
<tr>
<td>add</td>
<td></td>
<td>Adds devices to a device group and assigns the device a logical name</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>Lists all devices in a device group and any associated BCV devices</td>
</tr>
<tr>
<td>remove</td>
<td></td>
<td>Removes a device from a device group</td>
</tr>
<tr>
<td>rename</td>
<td></td>
<td>Renames a device in the device group</td>
</tr>
<tr>
<td>show</td>
<td></td>
<td>Shows detailed information about a device in the device group</td>
</tr>
<tr>
<td>symbcv</td>
<td></td>
<td>Performs support operations on BCV pairs</td>
</tr>
<tr>
<td>list</td>
<td></td>
<td>Lists BCV devices</td>
</tr>
<tr>
<td>Command</td>
<td>Argument</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>associate</td>
<td></td>
<td>Associates BCV devices to a device group – required to perform operations on the BCV device</td>
</tr>
<tr>
<td>disassociate</td>
<td></td>
<td>Disassociates BCV devices from a device group</td>
</tr>
<tr>
<td>associate –rdf</td>
<td></td>
<td>Associates remotely attached BCV devices to a SRDF device group</td>
</tr>
<tr>
<td>disassociate –rdf</td>
<td></td>
<td>Disassociates remotely attached BCV devices from an SRDF device group</td>
</tr>
</tbody>
</table>
EMC Change Tracker

The EMC Symmetrix Change Tracker software is also part of the base Solutions Enabler SYMCLI management offering. Change Tracker commands are used to measure changes to data on a Symmetrix volume or group of volumes. Change Tracker functionality is often used as a planning tool in the analysis and design of configurations that use the EMC SRDF and TimeFinder components to create copies of production data.

The Change Tracker command (symchg) is used to monitor the amount of changes being made to a group of hypervolumes. This command timestamps and marks specific volumes for tracking and maintains a bitmap to record which tracks have changed on those volumes. The bitmap can be interrogated to gain an understanding of how the data on the volume changes over time and to assess the locality of reference of applications.
EMC Symmetrix Remote Data Facility

The Symmetrix Remote Data Facility (SRDF) component of EMC Solutions Enabler extends the basic SYMCLI command set to enable users to manage SRDF. SRDF is a business continuity solution that provides a host-independent, mirrored data storage solution for duplicating production site data to one or more physically separated target Symmetrix systems. In basic terms, SRDF is a configuration of multiple Symmetrix systems whose purpose is to maintain multiple copies of logical volume data in more than one location.

SRDF replicates production or primary (source) site data to a secondary (target) site transparently to users, applications, databases, and host processors. The local SRDF device, known as the source (R1) device, is configured in a partner relationship with a remote target (R2) device, forming an SRDF pair. While the R2 device is mirrored with the R1 device, the R2 device is write-disabled to the remote host. After the R2 device synchronizes with its R1 device, the R2 device can be split from the R1 device at any time, making the R2 device fully accessible again to its host. After the split, the target (R2) device contains valid data and is available for performing business continuity tasks through its original device address.

SRDF requires configuration of specific source Symmetrix volumes (R1) to be mirrored to target Symmetrix volumes (R2). If the primary site is no longer able to continue processing when SRDF is operating in synchronous mode, data at the secondary site is current up to the last committed transaction. When primary systems are down, SRDF enables fast failover to the secondary copy of the data so that critical information becomes available in minutes. Business operations and related applications may resume full functionality with minimal interruption.

Figure 26 on page 100 illustrates a basic SRDF configuration where connectivity between the two Symmetrix is provided using ESCON, Fibre Channel, or Gigabit Ethernet. The connection between the R1 and R2 devices is through a logical grouping of devices called a remote adapter (RA) group. The RA group is independent of the device and composite groups defined; device and composite groups are discussed in “SRDF device groups and composite groups” on page 101.
Figure 26  Basic synchronous SRDF configuration

SRDF benefits

SRDF offers the following features and benefits:
- High data availability
- High performance
- Flexible configurations
- Host and application software transparency
- Automatic recovery from a component or link failure
- Significantly reduced recovery time after a disaster
- Increased integrity of recovery procedures
- Reduced backup and recovery costs
- Reduced disaster recovery complexity, planning, testing, etc.
- Supports Business Continuity across and between multiple databases on multiple servers and Symmetrix systems.

SRDF modes of operation

SRDF currently supports the following modes of operation:
- **Synchronous mode (SRDF/S)** provides real-time mirroring of data between the source Symmetrix system(s) and the target Symmetrix system(s). Data is written simultaneously to the cache of both systems in real time before the application I/O is completed, thus ensuring the highest possible data availability.
Data must be successfully stored in both the local and remote Symmetrix systems before an acknowledgment is sent to the local host. This mode is used mainly for metropolitan area network distances less than 200 km.

- **Asynchronous mode (SRDF/A)** maintains a dependent-write consistent copy of data at all times across any distance with no host application impact. Applications needing to replicate data across long distances historically have had limited options. SRDF/A delivers high-performance, extended-distance replication and reduced telecommunication costs while leveraging existing management capabilities with no host performance impact.

- **Adaptive copy mode** transfers data from source devices to target devices regardless of order or consistency, and without host performance impact. This is especially useful when transferring large amounts of data during data center migrations, consolidations, and in data mobility environments. Adaptive copy mode is the data movement mechanism of the Symmetrix Automated Replication (SRDF/AR) solution.

### SRDF device groups and composite groups

Applications running on Symmetrix systems normally utilize a number of Symmetrix devices. Therefore, any Symmetrix operation must ensure all related devices are operated upon as a logical group. Defining device or composite groups achieves this.

A device group or a composite group is a user-defined group of devices that SYMCLI commands can be executed upon. Device groups are limited to a single Symmetrix system and RA group (a.k.a. SRDF group). A composite group, on the other hand, can span multiple Symmetrix systems and RA groups. The device or composite group type may contain R1 or R2 devices and may contain various device lists for standard, BCV, virtual, and remote BCV devices. The `symdg`/`symld` and `symcg` commands are used to create and manage device and composite groups.

### SRDF consistency groups

An SRDF consistency group is a collection of devices defined by a composite group that has been enabled for consistency protection. Its purpose is to protect data integrity for applications that span multiple
RA groups and/or multiple Symmetrix systems. The protected applications may comprise multiple heterogeneous data resource managers across multiple host operating systems.

An SRDF consistency group uses PowerPath or Enginuity Consistency Assist (SRDF-ECA) to provide synchronous disaster restart with zero data loss. Disaster restart solutions that use consistency groups provide remote restart with short recovery time objectives. Zero data loss implies that all completed transactions at the beginning of a disaster will be available at the target.

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 run decision-support queries on this data become critical factors. The time required to quiesce or shut down the application for offline backup is no longer acceptable. SRDF consistency groups allow users to remotely mirror the largest data environments and automatically split off dependent-write consistent, restartable copies of applications in seconds without interruption to online service.

A consistency group is a composite group of SRDF devices (R1 or R2) that act in unison to maintain the integrity of applications distributed across multiple Symmetrix systems or multiple RA groups within a single Symmetrix. If a source (R1) device in the consistency group cannot propagate data to its corresponding target (R2) device, EMC software suspends data propagation from all R1 devices in the consistency group, halting all data flow to the R2 targets. This suspension, referred to as a “consistency group trip,” ensures that a dependent-write consistent R2 copy of the data up to the point in time that the consistency group tripped, exists.

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

- One or more R1 devices cannot propagate changes to their corresponding R2 devices
- The R2 device fails
- The SRDF directors on the R1 side or R2 side fail

In an automatic trip, the Symmetrix system completes the write to the R1 device, but indicates that the write did not propagate to the R2 device. EMC software intercepts the I/O and instructs the Symmetrix to suspend all R1 source devices in the consistency group from propagating any further writes to the R2 side. Once the suspension is
complete, writes to all of the R1 devices in the consistency group continue normally, but they are not propagated to the target side until normal SRDF mirroring resumes.

An explicit trip occurs when the command `symrdf -cg suspend` or `split` is invoked. Suspending or splitting the consistency group creates an on-demand, restartable copy of the data at the R2 target site. BCV devices that are synchronized with the R2 devices are then split after the consistency group is tripped, creating a second dependent-write consistent copy of the data. During the explicit trip, SYMCLI issues the command to create the dependent-write consistent copy, but may require assistance from PowerPath or SRDF-ECA if I/O is received on one or more R1 devices, or if the SYMCLI commands issued are abnormally terminated before the explicit trip.

An EMC consistency group maintains consistency within applications spread across multiple Symmetrix systems in an SRDF configuration, by monitoring data propagation from the source (R1) devices in a consistency group to their corresponding target (R2) devices as depicted in Figure 27. Consistency groups provide data integrity protection during a rolling disaster.

**Figure 27** SRDF consistency group

X = DBMS data
Y = Application data
Z = Logs
In the example depicted in Figure 27 on page 103, a consistency group containing volumes X, Y, and Z on the source Symmetrix is defined. This consistency group definition must contain all of the devices that need to maintain dependent-write consistency and that 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 some cases, the entire processing environment may be defined in a consistency group to ensure dependent-write consistency.

Next, the rolling disaster described previously begins, preventing the replication of changes from volume Z to the remote site. Since the predecessor log write to volume Z cannot be propagated to the remote Symmetrix system, a consistency group trip occurs.

The consistency group trip holds the write that could not be replicated along with all of the writes to the logically grouped devices. The writes are held by PowerPath on UNIX/Windows hosts and by IOS on mainframe hosts (or by ECA-RDA for both UNIX/Windows and mainframe hosts) long enough to issue two I/Os to all of the Symmetrix systems involved in the consistency group. The first I/O changes the state of the devices to a suspend-pending state.

The second I/O performs the suspend actions on the R1/R2 relationships 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. After the relationship is suspended, the completion of the predecessor write is acknowledged back to the issuing host. Furthermore, all writes that were held during the consistency group trip operation are released.

After the second I/O per Symmetrix completes, the I/O is 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).

When a complete failure occurs from this rolling disaster, the dependent-write consistency at the remote site is preserved. If a complete disaster does not occur and the failed links are activated again, the consistency group replication can be resumed. EMC recommends creating 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 terminology

This section describes various terms related to SRDF operations.

Suspend and resume operations

Practical uses of suspend and resume operations usually involve unplanned situations in which an immediate suspension of I/O between the R1 and R2 devices over the SRDF links is desired. In this way, data propagation problems can be stopped. When suspend is used with consistency groups, immediate backups can be performed off the R2 devices without affecting I/O from the local host application. I/O can then be resumed between the R1 and R2 and return to normal operation.

Establish and split operations

Normally, establish and split operations are used in planned situations in which use of the R2 copy of the data is desired without interfering with normal write operations to the R1 device. Splitting a point-in-time copy of data allows access to the data on the R2 device for various business continuity tasks. The ease of splitting SRDF pairs to provide exact database copies makes it convenient to perform scheduled backup operations, reporting operations, or new application testing from the target Symmetrix data while normal processing continues on the source Symmetrix system.

The R2 copy can also be used to test disaster recovery plans without manually intensive recovery drills, complex procedures, and application service interruptions. Upgrades to new versions can be tested or changes to actual code can be made without affecting the online production server. For example, modified server code can be run on the R2 copy of the data until the upgraded code runs with no errors before upgrading the production server.

In cases where an absolute real-time copy of the production data is not essential, users may choose to split the SRDF pair periodically and use the R2 copy for queries and report generation. The SRDF pair can be re-established periodically to provide incremental updating of data on the R2 device. The ability to refresh the R2 device periodically provides the latest information for data processing and reporting.
Failover and failback operations

Practical uses of failover and failback operations usually involve the need to switch business operations from the production site to a remote site (failover) or the opposite (failback). Once failover occurs, normal operations continue using the remote (R2) copy of synchronized application data. Scheduled maintenance at the production site is one example of where failover to the R2 site might be needed.

Testing of disaster recovery plans is the primary reason to temporarily fail over to a remote site. Traditional disaster recovery routines involve customized software and complex procedures. Offsite media must be either electronically transmitted or physically shipped to the recovery site. Time-consuming restores and the application of logs usually follow. SRDF failover/failback operations significantly reduce the recovery time by incrementally updating only the specific tracks that have changed; this accomplishes in minutes what might take hours for a complete load from dumped data volumes.

Update operation

The update operation allows users to resynchronize the R1 devices after a failover while continuing to run application and database services on the R2 devices. This function helps reduce the amount of time that a failback to the R1 side takes. The update operation is a subset of failover/failback functionality. Practical uses of the R1 update operation usually involve situations in which the R1 becomes almost synchronized with the R2 data before a failback, while the R2 side is still online to its host. The -until option, when used with update, specifies the target number of invalid tracks that are allowed to be out of sync before resynchronization to the R1 completes.

Concurrent SRDF

Concurrent SRDF means having two target R2 devices configured as concurrent mirrors of one source R1 device. Using a Concurrent SRDF pair allows the creation of two copies of the same data at two remote locations. When the two R2 devices are split from their source R1 device, each target site copy of the application can be accessed independently.
R1/R2 swap

Swapping R1/R2 devices of an SRDF pair causes the source R1 device to become a target R2 device and vice versa. Swapping SRDF devices allows the R2 site to take over operations while retaining a remote mirror on the original source site. Swapping is especially useful after failing over an application from the R1 site to the R2 site. SRDF swapping is available with Enginuity version 5567 or later.

Data mobility

Data mobility is an SRDF configuration that restricts SRDF devices to operating only in adaptive copy mode. This is a lower-cost licensing option that is typically used for data migrations. It allows data to be transferred in adaptive copy mode from source to target, and is not designed as a solution for DR requirements unless used in combination with TimeFinder.

Dynamic SRDF

Dynamic SRDF allows the creation of SRDF pairs from non-SRDF devices while the Symmetrix system is in operation. Historically, source and target SRDF device pairing has been static and changes required assistance from EMC personnel. This feature provides greater flexibility in deciding where to copy protected data.

Dynamic RA groups can be created in a SRDF switched fabric environment. An RA group represents a logical connection between two Symmetrix systems. Historically, RA groups were limited to those static RA groups defined at configuration time. However, RA groups can now be created, modified, and deleted while the Symmetrix system is in operation. This provides greater flexibility in forming SRDF-pair-associated links.

SRDF control operations

This section describes typical control operations that can be performed by the Solutions Enabler symrdf command.

Solutions Enabler SYMCLI SRDF commands perform the following basic control operations on SRDF devices:

- **establish** synchronizes an SRDF pair by initiating a data copy from the source (R1) side to the target (R2) side. This operation can be a full or incremental establish. Changes on the R2 devices are discarded by this process.
◆ **restore** resynchronizes a data copy from the target (R2) side to the source (R1) side. This operation can be a full or incremental restore. Changes on the R1 devices are discarded by this process.

◆ **split** stops mirroring for the SRDF pair(s) in a device group and write-enables the R2 devices.

◆ **swap** exchanges the source (R1) and target (R2) designations on the source and target volumes.

◆ **failover** switches data processing from the source (R1) side to the target (R2) side. The source side volumes (R1), if still available, are write-disabled.

◆ **failback** switches data processing from the target (R2) side to the source (R1) side. The target side volumes (R2), if still available, are write-disabled.

### Establishing an SRDF pair

The establishment of an SRDF pair initiates remote mirroring—the copying of data from the source (R1) device to the target (R2) device. SRDF pairs come into existence in two different ways:

◆ At configuration time through the pairing of SRDF devices. This is a static pairing configuration discussed earlier.

◆ Anytime during a dynamic pairing configuration in which SRDF pairs are created on demand.

A full establish (**symrdf establish -full**) is typically performed after an SRDF pair is initially configured and connected via the SRDF links. After the first full establish, users can perform an incremental establish, where the R1 device copies to the R2 device only the new data that was updated while the relationship was split or suspended.

To initiate an establish operation on all SRDF pairs in a device or composite group, all pairs must be in the split or suspended state. The **symrdf query** command can be used to check the state of SRDF pairs in a device or composite group.

When an establish operation is initiated, the system write-disables the R2 device to its host and merges the track tables. The merge creates a bitmap of the tracks that need to be copied to the target volumes, discarding the changes on the target volumes. When the establish operation is complete and the SRDF pairs are in the synchronized state. The R1 device and R2 device contain identical
data, and continue to do so until interrupted by administrative command or unplanned disruption. Figure 28 depicts SRDF establish and restore operations:

The establish operation may be initiated by any host connected to either Symmetrix system, provided that an appropriate device group has been built on that host. The following command initiates an incremental establish operation for all SRDF pairs in the device group named MyDevGrp:

```
symrdf -g MyDevGrp establish -noprompt
```

**Splitting an SRDF pair**

When read/write access to a target (R2) device is necessary, the SRDF pair can be split. When the split completes, the target host can access the R2 device for write operations. The R2 device contains valid data and is available for business continuity tasks or restoring data to the R1 device if there is a loss of data on that device.

While an SRDF pair is in the split state, local I/O to the R1 device can still occur. These updates are not propagated to the R2 device immediately. Changes on each Symmetrix system are tracked through bitmaps and are reconciled when normal SRDF mirroring operations are resumed. To initiate a split, an SRDF pair must already be in one of the following states:

- Synchronized
- Suspended
- R1 updated
SyncInProg (if the –symforce option is specified for the split — resulting in a set of R2 devices that are not dependent-write consistent and are not usable)

The split operation may be initiated from either host. The following command initiates a split operation on all SRDF pairs in the device group named MyDevGrp:

```
symrdf -g MyDevGrp split -noprompt
```

The symrdf split command provides exactly the same functionality as the symrdf suspend and symrdf rw_enable R2 commands together. Furthermore, the split and suspend operations have exactly the same consistency characteristics as SRDF consistency groups. Therefore, when SRDF pairs are in a single device group, users can split the SRDF pairs in the device group as shown previously and have restartable copies on the R2 devices. If the application data spans multiple Symmetrix systems or multiple RA groups, include SRDF pairs in a consistency group to achieve the same results.

Restoring an SRDF pair

When the target (R2) data must be copied back to the source (R1) device, the SRDF restore command is used (see Figure 28 on page 109). After an SRDF pair is split, the R2 device contains valid data and is available for business continuance tasks (such as running a new application) or for restoring data to the R1 device. Moreover, if the results of running a new application on the R2 device need to be preserved, moving the changed data and new application to the R1 device is another option.

Users can perform a full or incremental restore. A full restore operation copies the entire contents of the R2 device to the R1 device. An incremental restore operation is much faster because it copies only new data that was updated on the R2 device while the SRDF pair was split. Any tracks on the R1 device that changed while the SRDF pair was split are replaced with corresponding tracks on the R2 device. To initiate a restore, an SRDF pair must already be in the split state. The restore operation can be initiated from either host. The following command initiates an incremental restore operation on all SRDF pairs in the device group named MyDevGrp (add the –full option for a full restore).

```
symrdf -g MyDevGrp restore -noprompt
symrdf -g MyDevGrp restore -noprompt -full
```
The **restore** operation is complete when the R1 and R2 devices contain identical data. The SRDF pair is then in a synchronized state and may be re-established by initiating the following command:

```bash
symrdf -g MyDevGrp establish
```

**Failover and failback operations**

Having a synchronized SRDF pair allows users to switch data processing operations from the source site to the target site if operations at the source site are disrupted or if downtime must be scheduled for maintenance. This switchover from source to target is enabled through the use of the **failover** command. When the situation at the source site is back to normal, a **failback** operation is used to reestablish I/O communications links between source and target, resynchronize the data between the sites, and resume normal operations on the R1 devices as shown in Figure 29, which illustrates the **failover** and **failback** operations.

![Figure 29](ICO-IMG-000004)

The **failover** and **failback** operations relocate the processing from the source site to the target site or vice versa. This may or may not imply movement of data.

**Failover**

Scheduled maintenance or storage system problems can disrupt access to production data at the source site. In this case, a **failover** operation can be initiated from either host to make the R2 device read/write-enabled to its host. Before issuing the **failover**, all applications services on the R1 devices must be stopped. This is
because the \textit{failover} operation makes the R1 devices read-only. The following command initiates a failover on all SRDF pairs in the device group named MyDevGrp:

\texttt{symrdf -g MyDevGrp failover -noprompt}

In order to initiate a \textit{failover} operation, the SRDF pair must already be in one of the following states:

\begin{itemize}
  \item Synchronized
  \item Suspended
  \item R1 updated
  \item Partitioned (when invoking this operation at the target site)
\end{itemize}

Once initiated, a \textit{failover} operation:

\begin{itemize}
  \item Suspends data traffic on the SRDF links
  \item Write-disables the R1 devices
  \item Write-enables the R2 devices
\end{itemize}

\section*{Failback}

To resume normal operations on the R1 side, a \textit{failback} (R1 device takeover) operation is initiated. This means read/write operations on the R2 device must be stopped, and read/write operations on the R1 device must be started. When the \textit{failback} command is initiated, the R2 becomes read-only to its host, while the R1 becomes read/write-enabled to its host. The following command performs a \textit{failback} operation on all SRDF pairs in the device group named MyDevGrp:

\texttt{symrdf -g MyDevGrp failback -noprompt}

The SRDF pair must already be in one of the following states in order for a \textit{failback} operation to succeed:

\begin{itemize}
  \item Failed over
  \item Suspended and write-disabled at the source
  \item Suspended and not ready at the source
  \item R1 Updated
  \item R1 UpdInProg
\end{itemize}

Once initiated, a \textit{failback} operation:

\begin{itemize}
  \item Write-enables the R1 devices.
  \item Performs a track table merge to discard changes on the R1 devices.
  \item Transfers the changes on the R2 devices.
- Resumes traffic on the SRDF links.
- Write-disables the R2 devices.
EMC SRDF/Cluster Enabler solutions

EMC SRDF/Cluster Enabler (SRDF/CE) for MSCS is an integrated solution that combines SRDF and clustering protection over distance. EMC SRDF/CE provides disaster-tolerant capabilities that enable a cluster to span geographically separated Symmetrix systems. It operates as a software extension (MMC snap-in) to the Microsoft Cluster Service (MSCS).

SRDF/CE achieves this capability by exploiting SRDF disaster restart capabilities. SRDF allows the MSCS cluster to have two identical sets of application data in two different locations. When cluster services are failed over or failed back, SRDF/CE is invoked automatically to perform the SRDF functions necessary to enable the requested operation.

Figure 30 illustrates the hardware configuration of two, four-node, geographically distributed EMC SRDF/CE clusters using bidirectional SRDF.

![Figure 30 Geographically distributed four-node EMC SRDF/CE clusters](ICO-IMG-000005)
SRDF enhancements introduced with Enginuity 5875

EMC Enginuity 5875 is the latest intelligent, multitasking, preemptive storage operating environment (SOE) for the Symmetrix VMAX. As with previous Enginuity versions this release is devoted to storage operations and optimized for service levels required in high-end environments. And as expected, this Enginuity version on Symmetrix VMAX further advances the ability of EMC self-optimizing intelligence to deliver performance, array tiering, availability, and data integrity that now define advanced storage functionality.

Some of the more general SRDF enhancements made available with the Enginuity 5875 operating environment on VMAX storage arrays include:

- The ability to configure simultaneously multiple static SRDF groups
- The ability to throttle host write I/O response time up to a user-defined limit
- The ability to create a TimeFinder/Snap off an SRDF/A R2 device

In addition, the following new features were introduced with Enginuity 5875:

- **Concurrent SRDF/A** — Concurrent SRDF/A expands the SRDF multisite topology offering by allowing two separate asynchronous links from a Symmetrix VMAX to Symmetrix systems located at remote data centers. This configuration exploits the core benefits of SRDF/A for improved application response times while replicating at extended distances. Enginuity 5875 offers the flexibility to change Concurrent SRDF/S and SRDF/A disaster restart topologies to Concurrent SRDF/A and SRDF/A. This flexibility:
  - Allows you to meet performance goals during planned and known workload spike periods
  - Offers a new migration option for data center relocation
  - Provides additional disaster restart protection

- **Thick-to-thin migration with SRDF** — Thick-to-thin migration supports the migration of data between thick (standard) provisioned volumes on earlier generation Symmetrix systems to thin (virtually provisioned) volumes on new VMAX systems via SRDF replication. During replication, SRDF will detect tracks or blocks containing all zeros and remove them from the replication
stream, thereby allowing the reclamation of space during migration and the desired result of a thin migration target. Supported Enginuity combinations include:

- 5671 thick to 5875 thin
- 5773 thick to 5875 thin
- 5875 thick to 5875 thin
The SYMCLI TimeFinder component extends the basic SYMCLI command set to include TimeFinder or business continuance commands that perform control operations on device pairs within the TimeFinder environment.

Currently, the TimeFinder family consists of two separate and distinct software products along with several additional component options. The TimeFinder base replication products are:

- **TimeFinder/Clone** enables users to make 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, 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 TimeFinder BCV device.

- **TimeFinder/Snap** enables users to configure special devices in the Symmetrix array called *virtual devices* (VDEVs) and *save devices* (SAVEDEVs). 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 TimeFinder 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.

TimeFinder component options include:

- **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. TimeFinder/Mirror is a family component that works in Symmetrix environments running Enginuity 5773 and earlier. In environments running Enginuity 5874 and higher, all TimeFinder/Mirror scripts are executed in Clone Emulation mode.
IMPORTANT
Starting with Enginuity 5874, TimeFinder/Mirror functions are performed through TimeFinder/Clone software using a process called Clone Emulation. When running in Emulation Mode, TimeFinder/Clone transparently performs TimeFinder/Mirror commands and executes scripts that were written for Solutions Enabler up through version 6.5.2 running on Symmetrix arrays using Enginuity 5773 and earlier.

- **TimeFinder/Consistency Groups (TimeFinder/CG)** enables other TimeFinder family products to coordinate cross-volume and cross-system consistency to ensure application restartability. This option for Symmetrix arrays creates multivolume sets of point-in-time copies at the same instant, ensuring that as a set the copies are consistent and restartable, even when spread across multiple Symmetrix volumes and data is spread across multiple arrays, without quiescing or shutting down to ensure that all devices are copied at the same point in time.

- **TimeFinder/Exchange Integration Module (TimeFinder/EIM)** automates and simplifies the process of creating and managing TimeFinder replications of a Microsoft Windows Exchange Server environment.

- **TimeFinder/SQL Integration Module (TimeFinder/SIM)** automates (TimeFinder/SIM) simplifies the process of creating and managing TimeFinder replications of a Microsoft Windows SQL Server environment.

- **Duplicate TimeFinder/Snap** offers the capability to capture TimeFinder/Snap replicas from another TimeFinder/Snap point-in-time copy. This functionality is targeted mainly at SAP and database environments where copies of production environments are repurposed for testing, QA, or development. Work can proceed against an existing TimeFinder/Snap while duplicate copies can be created for additional downstream processes or checkpoint backups. With Enginuity 5875, snap copies can be taken from another snap source, adding even more disk space savings and flexibility through this track sharing technology.

The commands that comprise the TimeFinder component technologies of the EMC Solutions Enabler are: `symclone`, `symsnap`, `symmir`, `symbcv`, and `symioctl`. The TimeFinder/Clone command,
**symclone**, is used to create a point-in-time copy of a source device on nonstandard device pairs (such as standard/standard and BCV/BCV). The TimeFinder/Snap command, **symsnap**, is used to create virtual device copy sessions between a source device and multiple virtual target devices. These virtual devices only store pointers to changed data blocks from the source device, rather than a full copy of the data.

Base component commands such as **symmir** and **symbcv** are used to perform a wide spectrum of monitor and control operations on Symmetrix standard/BCV device pairs within a TimeFinder environment. The **symioctl** command is used to send I/O control commands to a specified database server.

Configuring and controlling remote BCV pairs requires the EMC SRDF business continuity software that was discussed earlier. The combination of TimeFinder with SRDF provides for multiple local and remote copies of production data.

**Figure 31** illustrates application usage for a TimeFinder/Mirror configuration in a Symmetrix system.

---

**Figure 31**  
EMC Symmetrix configured with standard volumes and BCVs

---

**TimeFinder/Clone operations**

TimeFinder/Clone functionality is controlled via copy sessions, which pair source and target devices. Sessions are maintained on the Symmetrix system and can be queried to verify the current state of the device pairs. A copy session must first be created to define and set up the TimeFinder/Clone devices. The session is then activated, enabling the target device to be accessed by its host. When the information is no longer needed, the session can be terminated.
TimeFinder/Clone operations are controlled from the host by using the symclone command to create, activate, restore, recreate, set mode, split, establish, and terminate copy sessions.

Figure 32 illustrates a clone session where the controlling host creates a clone copy of standard device DEV001 on target device DEV005.

Figure 32  A TimeFinder/Clone copy of a standard device

TimeFinder/Clone operations are controlled from the host by using the symclone command to create, activate, restore, recreate, set mode, split, establish, and terminate clone sessions. These operations are discussed in great detail in Chapter 4, “EMC TimeFinder Family of Products.”

TimeFinder/Snap operations

Much like TimeFinder/Clone, TimeFinder/Snap functionality is managed through copy sessions, which pair source and target virtual devices. Sessions are maintained on the Symmetrix system and can be queried to verify the current state of the devices. A copy session must be created first—this defines the devices that will be used in the copy operation. Once the session is subsequently activated, the target virtual devices then become accessible to its host. When the information is no longer needed, the session can be terminated. In addition, data can also be restored from the virtual devices back to
the source devices. Figure 33 illustrates a virtual copy session where
the controlling host creates a copy of standard device DEV001 on
target device VDEV005.

As illustrated in Figure 33, a virtual device is a host-accessible device
that contains a collection of pointers to unchanged data and to save
data. When you activate a snap pair, disk space is consumed (on
SAVE devices) only when data is written to the source device or
VDEV, and then only the space required to accommodate pre-update
data from the source tracks that changed.

Pointers on the virtual device are initialized to point to tracks on the
source device. The first time new data is written to a track on the
source device, the original track data is copied to a SAVE device, and
the pointer on the virtual device is changed to point to that original
data. When a track is written to a VDEV, it also gets copied to the
SAVE device and the pointer on the VDEV is changed to point to the
SAVE device.
TimeFinder/Snap operations are controlled from the host by using the `symsnap` command to create, activate, terminate, and restore the snap copy sessions. These operations are discussed in great detail in Chapter 4, “EMC TimeFinder Family of Products.”

**Note:** Starting with Solutions Enabler version 5.4, TimeFinder operations using the SYMCLI `symmir`, `symclone`, and `symsnap` commands support composite groups (`-cg`) or devices in a composite group, as well as device groups (`-g`) and devices within a device group.

---

**TimeFinder/Mirror operations**

Symmetrix TimeFinder/Mirror is essentially a business continuance solution that allows the use of special business continuance volume (BCV) Symmetrix devices. Copies of data from a standard Symmetrix device (which are online for regular I/O operations from the host) are sent and stored on BCV devices to mirror the primary data. Uses for the BCV copies can include backup, restore, decision support, and applications testing. Each BCV device has its own host address, and is configured as a stand-alone Symmetrix device.

A Business Continuance sequence first involves associating and establishing the BCV device as a mirror of a specific standard Symmetrix device. As a result, the BCV device becomes inaccessible (Not Ready) using its original device address while it is in an established pair. Once the BCV device is synchronized, you can separate (split) it from the standard device with which it is paired, thereby making it available again to its host for backup or other host processes through its original device address. After host processing on the BCV device is complete, the BCV may again be mirrored to a standard Symmetrix device — either the same device with which it was previously paired, or with a different device.

TimeFinder/Mirror operations are performed using the `symbcv` and the `symmir` commands. The `symbcv` command enables you to perform the following tasks on device pairs:

- Associate a BCV device with a device group or a composite group.
- Associate a BCV device or all BCV devices to the same group containing the standard devices.
- Disassociate a BCV device from a device group or a composite group.
Disassociate the associated BCV device(s) from the specified group.
List all BCV devices in the Symmetrix array.
Move a BCV device from one group to another.
Copy a BCV device from one group to another.
Remove all BCV devices from the specified device group.

On the other hand, the symmir command enables you to perform the following mirroring tasks on standard-BCV pairs:

- Fully establish a BCV pair.
- Incrementally establish a BCV pair.
- Concurrently establish BCV pairs.
- Split a BCV pair.
- Fully restore from a BCV device.
- Incrementally restore from a BCV device.
- Specify a preferred attachment of a BCV to or from the standard (optional tasks).
- Cancel the existing relationship between the specified standard and BCV device(s).

These operations are discussed in great detail in Chapter 4, “EMC TimeFinder Family of Products.”

EMC Replication Manager

EMC Replication Manager is an EMC software application that dramatically simplifies the management and use of disk-based replications to improve the availability of users’ mission-critical data and rapid recovery of that data in case of corruption.

Replication Manager makes it easy to create point-in-time, disk-based replicas of applications, file systems, or logical volumes residing on existing storage arrays. Specifically, Replication Manager can:

- Create point-in-time replicas of production data in seconds.
- Facilitate quick, frequent, and non-destructive backups from replicas.
- Mount replicas to alternate hosts to facilitate offline processing (for example, decision-support services, integrity checking, and offline reporting).
- Restore deleted or damaged information quickly and easily from a disk replica.
Set the retention period for replicas so that storage is made available automatically.

Replication Manager helps users manage replicas as if they were tape cartridges in a tape library unit. The creation of replicas may be scheduled or replicas may be created on demand, with predefined expiration periods and automatic mounting to alternate hosts for backups or scripted processing. Different levels of access (assigned to individual users) ensure system and replica integrity.

Replicas created by Replication Manager can be stored on Symmetrix TimeFinder/Mirrors, Clones, or Snapshots (VDEVs); CLARiiON clones or snapshots; Invista® clones, Celerra® SnapSure™ local snapshots, or Celerra Replicator™ remote snapshots. Replication Manager also allows you to perform local and remote replications using TimeFinder, Symmetrix Remote Data Facility (SRDF), SAN Copy™, Navisphere®, Celerra iSCSI, and/or replicas of MirrorView™/S secondaries using SnapView™/Snap and SnapView/Clone replication technologies where they are appropriate. Additionally, Replication Manager automatically controls the complexities associated with creating, mounting, restoring, and expiring replicas of data.

There are several phases to the recovery process and Replication Manager helps to shorten each of these phases. The following list describes how Replication Manager shortens each of the phases in the data recovery process:

- **Latent Error phase**
  An error occurs in the data, but is not immediately detected. Replication Manager can provide separate replicas to verify the integrity of the data and actively search for errors. Proactive data scrubbing, a process by which Replication Manager creates a point-in-time replica and automated scripts scrub the data to find errors, can reduce or eliminate the latent error phase.

- **Evaluation phase**
  After an error is detected, you must evaluate the data and determine the best way to fix the error. You might choose to perform a surgical repair, making manual changes to the database to fix the error. Or, you might decide to restore the database from a replica and recover that database by applying the logs. You can shorten the evaluation process by creating a replica of the damaged database and using the replica to perform the evaluation, rather than using the production data.
**Surgery phase**

If you decide to perform a surgical repair, you can create a replica of the current data before the surgery. If something goes wrong during the manual database edit, you can restore the replica and attempt the surgery again. Restoring a replica is much faster than restoring from tape after a failed attempt to surgically repair the database. When the system is down, it is important to save as much time as possible.

**Restore and Roll Forward phase**

If you decide to restore the data from a replica, you can check each replica and choose the most recent replica that does not have the latent error. After that replica has been restored, use logs to roll the database forward, and then manually restart the database. You cannot perform the same validity check before restoring a tape.

Replication Manager can shorten the overall recovery process. Most other products focus on shortening only the Restore phase, while Replication Manager offers a complete solution that can save time throughout all phases of the recovery process.

Replication Manager uses Symmetrix API (SYMAPI) Solutions Enabler software and interfaces to the storage array’s native software to manipulate the supported disk arrays. (The Replication Manager software utilizes Java-based client-server architecture.) Replication Manager also offers a logical view of the production data and corresponding replicas; replicas are managed and controlled with the easy-to-use Replication Manager console.
EMC Storage Resource Management

The Storage Resource Management (SRM) component of EMC Solutions Enabler extends the basic SYMCLI command set to include SRM commands that allow users to discover and examine attributes of various objects on a host or in the EMC storage enterprise. SYMCLI commands support SRM in the following areas:

- Data objects and files
- Relational databases
- File systems
- Logical volumes and volume groups
- Performance statistics

SRM allows users to examine the mapping of storage devices and the characteristics of data files and objects. These commands allow the examination of relationships between extents and data files or data objects, and how they are mapped on storage devices. Frequently, SRM commands are used with TimeFinder and SRDF to create point-in-time copies for backup and restart.

Figure 34 outlines the process of how SRM commands are used with TimeFinder in a database environment.
EMC Solutions Enabler with a valid license for TimeFinder and SRM is installed on the host. In addition, the host must also have PowerPath or use ECA, and must be utilized with a supported DBMS system. When splitting a BCV, the system must perform housekeeping tasks that may require a few seconds on a busy Symmetrix system. These tasks involve a series of steps (shown in Figure 34 on page 126) that result in the separation of the BCV from its paired standard:

1. Using the SRM base mapping commands, first query the Symmetrix system to display the logical-to-physical mapping information about any physical device, logical volume, file, directory, and/or file system.

2. Using the database mapping command, query the Symmetrix to display physical and logical database information.

3. Next, use the database mapping command to translate:
   - The devices of a specified database into a device group or a consistency group, or
   - The devices of a specified table space into a device group or a consistency group.

4. The BCV is split from the standard device.

Table 4 lists the SYMCLI commands used to examine the mapping of data objects.

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>symrslv</td>
<td>pd</td>
<td>Displays logical to physical mapping information about any physical device.</td>
</tr>
<tr>
<td></td>
<td>lv</td>
<td>Displays logical to physical mapping information about a logical volume.</td>
</tr>
<tr>
<td></td>
<td>file</td>
<td>Displays logical to physical mapping information about a file.</td>
</tr>
<tr>
<td></td>
<td>dir</td>
<td>Displays logical to physical mapping information about a directory.</td>
</tr>
<tr>
<td></td>
<td>fs</td>
<td>Displays logical to physical mapping information about a file system.</td>
</tr>
</tbody>
</table>

SRM commands allow users to examine the host database mapping and the characteristics of a database. The commands provide listings and attributes that describe various databases, their structures, files, table spaces, and user schemas. Typically, the database commands work with Oracle, Informix, SQL Server, Sybase, Microsoft Exchange, SharePoint Portal Server, and DB2 LUW database applications.
Table 5 lists the SYMCLI commands used to examine the mapping of database objects.

**Table 5  Data object mapping commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Action</th>
</tr>
</thead>
</table>
| symrdb  | list     | Lists various physical and logical database objects:  
|         |          | - Current relational database instances available  
|         |          | - Table spaces, tables, files, or schemas of a database  
|         |          | - Files, segments, or tables of a database table space or schema |
| show    |          | Shows information about a database object: table space, tables, file, or schema of a database, File, segment, or a table of a specified table space or schema |
| rdb2dg  |          | Translates the devices of a specified database into a device group. |
| rdb2cg  |          | Translates the devices of a specified table space into a composite group or a consistency group. |
| tbs2cg  |          | Translates the devices of a specified table space into a composite group. Only data database files are translated. |
| tbs2dg  |          | Translates the devices of a specified table space into a device group. Only data database files are translated. |

The SYMCLI file system SRM command allows users to investigate the file systems that are in use on the operating system. This command provides listings and attributes that describe file systems, directories, and files, and their mapping to physical devices and extents.

Table 6 lists the SYMCLI command that can be used to examine a file system mapping.

**Table 6  File system SRM commands to examine file system mapping**

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>symhostfs</td>
<td>list</td>
<td>Displays a list of file systems, files, or directories.</td>
</tr>
<tr>
<td></td>
<td>show</td>
<td>Displays more detail information about a file system or file system object.</td>
</tr>
</tbody>
</table>
SYMCLI logical volume SRM commands allow users to map logical volumes to display a detailed view of the underlying storage devices. Logical volume architecture defined by a Logical Volume Manager (LVM) is a means for advanced applications to improve performance by the strategic placement of data.

Table 7 lists the SYMCLI commands that can be used to examine a logical volume mapping.

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>symvg</td>
<td>deport</td>
<td>Deports a specified volume group so it can be imported later.</td>
</tr>
<tr>
<td></td>
<td>import</td>
<td>Imports a specified volume group.</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>Displays a list of volume groups defined on the host system by the logical volume manager.</td>
</tr>
<tr>
<td></td>
<td>rescan</td>
<td>Rescans all the volume groups.</td>
</tr>
<tr>
<td></td>
<td>show</td>
<td>Displays more detail information about a volume group.</td>
</tr>
<tr>
<td>vg2cg</td>
<td></td>
<td>Translates volume groups to composite groups.</td>
</tr>
<tr>
<td>vg2dg</td>
<td></td>
<td>Translates volume groups to device groups.</td>
</tr>
<tr>
<td>symlv</td>
<td>list</td>
<td>Displays a list of logical volumes on a specified volume group.</td>
</tr>
<tr>
<td></td>
<td>show</td>
<td>Displays detail information (including extent data) about a logical volume.</td>
</tr>
</tbody>
</table>

SRM performance statistics commands allow users to retrieve statistics about a host’s CPU, disk, and memory.

Table 8 lists the statistics commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>symbost</td>
<td>show</td>
<td>Displays host configuration information.</td>
</tr>
<tr>
<td></td>
<td>stats</td>
<td>Displays performance statistics.</td>
</tr>
</tbody>
</table>
EMC Foundation Products

EMC PowerPath

EMC PowerPath is host-based software that works with networked storage systems to intelligently manage I/O paths. PowerPath manages multiple paths to a storage array. Supporting multiple paths enables recovery from path failure because PowerPath automatically detects path failures and redirects I/O to other available paths. PowerPath also uses sophisticated algorithms to provide dynamic load balancing for several kinds of path management policies that the user can set. With the help of PowerPath, systems administrators are able to ensure that applications on the host have highly available access to storage and perform optimally at all times.

A key feature of path management in PowerPath is dynamic, multipath load balancing. Without PowerPath, an administrator must statically load balance paths to logical devices to improve performance. For example, based on current usage, the administrator might configure three heavily used logical devices on one path, seven moderately used logical devices on a second path, and 20 lightly used logical devices on a third path. As I/O patterns change, these statically configured paths may become unbalanced, causing performance to suffer. The administrator must then reconfigure the paths, and continue to reconfigure them as I/O traffic between the host and the storage system shifts in response to usage changes.

Designed to use all paths concurrently, PowerPath distributes I/O requests to a logical device across all available paths, rather than requiring a single path to bear the entire I/O burden. PowerPath can distribute the I/O for all logical devices over all paths shared by those logical devices, so that all paths are equally burdened. PowerPath load balances I/O on a host-by-host basis, and maintains statistics on all I/O for all paths. For each I/O request, PowerPath intelligently chooses the least-burdened available path, depending on the load-balancing and failover policy in effect. In addition to improving I/O performance, dynamic load balancing reduces management time and downtime because administrators no longer need to manage paths across logical devices. With PowerPath, configurations of paths and policies for an individual device can be changed dynamically, taking effect immediately, without any disruption to the applications.

PowerPath provides the following features and benefits:
◆ **Multiple paths, for higher availability and performance** — PowerPath supports multiple paths between a logical device and a host bus adapter (HBA, a device through which a host can issue I/O requests). Having multiple paths enables the host to access a logical device even if a specific path is unavailable. Also, multiple paths can share the I/O workload to a given logical device.

◆ **Dynamic multipath load balancing** — Through continuous I/O balancing, PowerPath improves a host’s ability to manage heavy I/O loads. PowerPath dynamically tunes paths for performance as workloads change, eliminating the need for repeated static reconfigurations.

◆ **Proactive I/O path testing and automatic path recovery** — PowerPath periodically tests failed paths to determine if they are available. A path is restored automatically when available, and PowerPath resumes sending I/O to it. PowerPath also periodically tests available but unused paths, to ensure they are operational.

◆ **Automatic path failover** — PowerPath automatically redirects data from a failed I/O path to an alternate path. This eliminates application downtime; failovers are transparent and non-disruptive to applications.

◆ **Enhanced high availability cluster support** — PowerPath is particularly beneficial in cluster environments, as it can prevent interruptions to operations and costly downtime. PowerPath’s path failover capability avoids node failover, maintaining uninterrupted application support on the active node in the event of a path disconnect (as long as another path is available).

◆ **Consistent split** — PowerPath allows users to perform TimeFinder consistent splits by suspending device writes at the host level for a fraction of a second while the foreground split occurs. PowerPath software provides suspend-and-resume capability that avoids inconsistencies and restart problems that can occur if a database-related BCV is split without first quiescing the database.

◆ **Consistency Groups** — Consistency groups are a composite group of Symmetrix devices specially configured to act in unison to maintain the integrity of a database distributed across multiple SRDF arrays controlled by an open systems host computer.
EMC Open Replicator

EMC Open Replicator enables distribution and/or consolidation of remote point-in-time copies between EMC Symmetrix and qualified storage systems such as the EMC CLARiiON storage arrays. By leveraging the high-end Symmetrix storage architecture, Open Replicator offers unmatched deployment flexibility and massive scalability.

Open Replicator can be used to provide solutions to business processes that require high-speed data mobility, remote vaulting and data migration. Specifically, Open Replicator enables customers to:

- Rapidly copy data between Symmetrix, CLARiiON and third-party storage arrays.
- Perform online migrations from qualified storage to Symmetrix arrays with minimal disruption to host applications.
- Push a point-in-time copy of applications from Symmetrix arrays to a target volume on qualified storage arrays with incremental updates.
- Copy from source volumes on qualified remote arrays to Symmetrix volumes.

Open Replicator is tightly integrated with the EMC TimeFinder and SRDF family of products, providing enterprises with highly flexible and lower-cost options for remote protection and migration. Open Replicator is ideal for applications and environments where economics and infrastructure flexibility outweigh RPO and RTO requirements. Open Replicator enables businesses to:

- Provide a cost-effective and flexible solution to protect lower-tier applications.
- Reduce TCO by pushing or pulling data from Symmetrix systems to other qualified storage arrays in conventional SAN/WAN environments.
- Create remote point-in-time copies of production applications for many ancillary business operations such as data vaulting.
- Obtain cost-effective application restore capabilities with minimal RPO/RTO impact.
- Comply with industry policies and government regulations.
EMC Virtual Provisioning

Virtual Provisioning™ (commonly known as thin provisioning) was released with the 5773 Enginuity operating environment. Virtual Provisioning allows for storage to be allocated/accessed on-demand from a pool of storage servicing one or many applications. This type of approach has multiple benefits:

- Enables LUNs to be “grown” into over time with no impact to the host or application as space is added to the thin pool
- Only delivers space from the thin pool when it is written to, that is, on-demand. Overallocated application components only use space that is written to — not requested.
- Provides for thin-pool wide striping and for the most part relieves the storage administrator of the burden of physical device/LUN configuration

Virtual Provisioning introduces two new devices to the Symmetrix: a thin device and a data device.

---

Thin device

A thin device is a “host accessible device” that has no storage directly associated with it. Thin devices have pre-configured sizes and appear to the host to have that exact capacity. Storage is allocated in chunks when a block is written to for the first time. Zeros are provided to the host for data that is read from chunks that have not yet been allocated.

---

Data device

Data devices are specifically configured devices within the Symmetrix that are containers for the written-to blocks of thin devices. Any number of data devices may comprise a data device pool. Blocks are allocated to the thin devices from the pool on a round robin basis. The allocation block size is 768K.

Figure 35 on page 134 depicts the components of a Virtual Provisioning configuration:
New Symmetrix VMAX Virtual Provisioning features

Solutions Enabler 7.1 and Enginuity 5874 SR1 introduce two new features to Symmetrix Virtual Provisioning — thin pool write rebalancing and zero space reclamation. Thin pool write rebalancing provides the ability to automatically rebalance allocated extents on data devices over the entire pool when new data devices are added. Zero space reclamation allows users to reclaim space from tracks of data devices that are all zeros.

Thin pool write rebalance

Thin pool write rebalancing for Virtual Provisioning pools extends the functionality of the Virtual Provisioning feature by implementing a method to normalize the used capacity levels of data devices within a Virtual data pool after new data drives are added or existing data drives are drained. This feature introduces a background optimization task to scan the used capacity levels of the data devices within a virtual pool and perform movements of multiple track groups from the most utilized pool data devices to the least used pool.
data devices. The process can be scheduled to run only when changes to virtual pool composition make it necessary and user controls exist to specify what utilization delta will trigger track group movement.

Zero space reclamation

Zero space reclamation or Virtual Provisioning space reclamation provides the ability to free, also referred to as "de-allocate," storage extents found to contain all zeros. This feature is an extension of the existing Virtual Provisioning space de-allocation mechanism. Previous versions of Enginuity and Solutions Enabler allowed for reclaiming allocated (reserved but unused) thin device space from a thin pool. Administrators now have the ability to reclaim both allocated/unwritten extents as well as extents filled with host-written zeros within a thin pool. The space reclamation process is nondisruptive and can be executed with the targeted thin device ready and read/write to operating systems and applications.

When the space reclamation process is initiated, a back-end disk director (DA) task that will examine the allocated thin device extents on specified thin devices is started. A thin device extent is 768 KB (or 12 tracks) in size and is the default unit of storage at which allocations occur. For each allocated extent, all 12 tracks will be brought into Symmetrix cache and examined to see if they contain all zero data. If the entire extent contains all zero data, the extent will be de-allocated and added back into the pool, making it available for a new extent allocation operation. An extent that contains any non-zero data is not reclaimed.

New Symmetrix VMAX TimeFinder/Clone features

Solutions Enabler 7.1 and Enginuity 5874 SR1 introduce the ability to clone from thick to thin devices using TimeFinder/Clone. Thick-to-thin TimeFinder/Clone allows application data to be moved from standard Symmetrix volumes to virtually provisioned storage within the same array. For some workloads virtually provisioned volumes offer advantages with allocation utilization, ease of use and performance through automatic wide striping. Thick-to-thin TimeFinder/Clone provides an easy way to move workloads that benefit from Virtual Provisioning into that storage paradigm. Migration from thin devices back to fully provisioned devices is also possible. The source and target of the migration may be of different protection types and disk technologies, offering versatility with
protections schemes and disk tier options. Thick-to-thin TimeFinder Clone will not disrupt hosts or internal array replication sessions during the copy process.
EMC Fully Automated Storage Tiering (FAST)

With the release of Enginuity 5874, EMC now offers the first generation of Fully Automated Storage Tiering technology. EMC Symmetrix VMAX Fully Automated Storage Tiering (FAST) for standard provisioned environments automates the identification of data volumes for the purposes of allocating or re-allocating application data across different performance tiers within an array. FAST proactively monitors workloads at the volume (LUN) level in order to identify “busy” volumes that would benefit from being moved to higher-performing drives. FAST will also identify less “busy” volumes that could be relocated to higher-capacity drives, without existing performance being affected. This promotion/demotion activity is based on policies that associate a storage group to multiple drive technologies, or RAID protection schemes, based on the performance requirements of the application contained within the storage group. Data movement executed during this activity is performed nondisruptively, without affecting business continuity and data availability.

The primary benefits of FAST include:

- Automating the process of identifying volumes that can benefit from Enterprise Flash Drives and/or that can be kept on higher-capacity, less-expensive drives without impacting performance
- Improving application performance at the same cost, or providing the same application performance at lower cost. Cost is defined as space, energy, acquisition, management and operational expense.
- Optimizing and prioritizing business applications, which allows customers to dynamically allocate resources within a single array
- Delivering greater flexibility in meeting different price/performance ratios throughout the lifecycle of the information stored

Management and operation of FAST are provided by SMC, as well as the Solutions Enabler Command Line Interface (SYMCLI). In addition, detailed performance trending, forecasting, alerts, and resource utilization are provided through Symmetrix Performance Analyzer (SPA). Ionix ControlCenter provides the capability for advanced reporting and analysis to be used for chargeback and capacity planning.
FAST VP with Enginuity 5875 extends current FAST capabilities to include both thick (standard) devices and thin (virtually provisioned) devices. Building on the original version of FAST, EMC now offers sub-LUN data movement for thin devices providing increased capacity utilization.

Typically, only a small portion of any LUN is actively supporting workload I/O activity. Providing sub-LUN data movement at a much more granular level (smaller pieces) means production workloads can enjoy the benefits of improved performance and improved capacity utilization. For example, sub-LUN FAST can experience the benefits of improved performance from placing data on Enterprise Flash Drives (EFDs) while using fewer EFDs. Since a majority of the data is low activity data (due to workload skew), it can be placed on the FC and SATA drives, preserving performance while saving cost. Providing sub-LUN data movement at a much more granular level, called extents, allows FAST to be more responsive to changes in the production workload activity, thereby:

- Improving performance
- Efficiently utilizing capacity
- Requiring fewer EFDs in the system
- Allowing more data to be placed on SATA drives
Over time, a database environment can encounter any number of problems, including power interruptions, storage media failure, and application abends. All of these can result in database failure, and each failure scenario requires a different recovery action. To prevent such problems from resulting in catastrophic data loss, a set of tools that can be used to back up and restore databases are provided with DB2 for Linux, UNIX, and Windows (LUW).

This chapter is designed to introduce you to the back up and recovery tools that are available with DB2. This chapter is also designed to show you how to both back up a database on a regular basis and restore a database if it becomes damaged or corrupted, using the tools provided with DB2 LUW. Topics covered include:

- Database recovery concepts............................................................ 140
- Performing a crash recovery operation ........................................ 148
- Back up and recovery ................................................................. 152
- Backing up a database with split mirroring................................. 174
Database recovery concepts

The concept of backing up a database is the same as that of backing up any other set of data files: you make a copy of the data and store it on a different medium where it can be accessed in the event the original becomes damaged or destroyed. The simplest way to back up a database is to shut it down to ensure that no further transactions are processed and then back it up using the Backup utility provided with DB2. Once a back up image has been created, you can use it to rebuild the database if for some reason it becomes damaged or corrupted.

The process of rebuilding a database is known as recovery, and three types of recovery are available with DB2:
- Crash recovery
- Version recovery
- Roll-forward recovery

Crash recovery

When an event or condition that causes a database or the DB2 Database Manager to end abnormally takes place, one or more transaction failures may result. Conditions that can cause transaction failure include the following:
- A power failure at the server where the DB2 Database Manager is running
- A serious operating system error
- A hardware failure such as memory corruption, disk failure, CPU failure, or network failure

When a transaction failure takes place, all work done by partially completed transactions that have not yet been externalized to the database is lost. As a result, the database may be left in an inconsistent state (and therefore will be unusable). Crash recovery is the process used to return such a database to a consistent and usable state. Crash recovery is performed by using information stored in the transaction log files to complete any committed transactions that were in memory (but had not yet been externalized to storage) when the failure occurred, roll back any incomplete transactions found, and purge any uncommitted transactions from memory. Once a database
is returned to a consistent and usable state, it has attained what is known as a “point of consistency.” Crash recovery is illustrated in Figure 36.

![Figure 36 Crash recovery](image.png)

A crash recovery operation is initiated by executing the **RESTART DATABASE** command.

**Note:** By default, a DB2 database is configured such that crash recovery will automatically be performed if an application or user attempts to establish a connection to it while it is in an inconsistent state. (This behavior can be changed by assigning the value **OFF** to a database’s **autorestart** configuration parameter.)
Version recovery is the process used to return a database to the state it was in at the time a particular back up image was made. Version recovery is performed by replacing the current version of a database with a previous version, using a copy that was made with a back up operation—the entire database is rebuilt using a back up image that was created earlier. Unfortunately, when a version recovery is performed, all changes made to the database since the back up image used was created are lost. Version recovery is illustrated in Figure 37.

Figure 37 Version recovery

A version recovery operation is initiated by executing the RESTORE DATABASE command; database back up images needed for version recovery operations are generated by executing the BACKUP DATABASE command.
Roll-forward recovery

Roll-forward recovery takes version recovery one step further by rebuilding a database or one or more individual table spaces using a backup image and replaying information stored in transaction log files to return the database/table spaces to the state they were in at an exact point in time. In order to perform a roll-forward recovery operation, you must have archival logging enabled, you must have either a full backup image of the database or a complete set of table space backup images available, and you must have access to all archived log files that have been created since the backup image(s) were made. Roll-forward recovery is illustrated in Figure 38.

A roll-forward recovery operation is initiated by executing the `ROLLFORWARD DATABASE` command.
Recoverable and nonrecoverable databases

Although any DB2 database can be recovered from a back up image, whether a database is considered recoverable or nonrecoverable is determined by the values of the database's `logarchmeth1` and `logarchmeth2` configuration parameters; when both of these configuration parameters are set to `OFF`—which is the default—circular logging is used, and the database is considered nonrecoverable. When either of the configuration parameters are assigned a value other than, `OFF` the database is considered recoverable. Crash recovery and version recovery operations can be performed on nonrecoverable databases; crash recovery, version recovery, and roll-forward recovery are possible when a database is recoverable. Other differences between recoverable and nonrecoverable databases can be seen in Table 9.

<table>
<thead>
<tr>
<th>Recoverable database</th>
<th>Nonrecoverable database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archive logging is used</td>
<td>Circular logging is used</td>
</tr>
<tr>
<td>The database can be backed up at any time, regardless of whether applications are connected to it and transactions are in progress.</td>
<td>The database can be backed up only when all connections to it have been terminated.</td>
</tr>
<tr>
<td>The entire database can be backed up, or individual table spaces can be backed up. Table spaces can also be restored independently.</td>
<td>The entire database must be backed up; table space-level back up operations are not supported.</td>
</tr>
<tr>
<td>A damaged database can be returned to the state it was in at any point in time; crash recovery, version recovery, and roll-forward recovery are supported.</td>
<td>A damaged database can be returned only to the state it was in at the time the last back up image was taken; only crash recovery and version recovery are supported.</td>
</tr>
</tbody>
</table>

**Note:** Some relational database management systems refer to nonrecoverable databases as *restartable* databases. Typically, the recovery options available for a restartable database are the same as for a nonrecoverable database.

The decision of whether a database should be recoverable or nonrecoverable is based on several factors:

- If a database is used to support read-only operations, it can be nonrecoverable; because no transactions will be logged, roll-forward recovery is not necessary.
If relatively few changes will be made to a database, and if all changes made can be easily re-created, it may be desirable to leave the database nonrecoverable.

If a large amount of changes will be made to a database, or if it would be difficult and time-consuming to re-create all changes made, the database should be recoverable.
Online versus offline back up and recovery

From a back up and recovery perspective, a database is considered to be either online or offline: when a database is offline, other applications and users cannot gain access to it; when a database is online, just the opposite is true. Back up operations can only be performed against a nonrecoverable database (that is, a database that has been configured to use circular logging) while it is offline; recoverable databases (i.e., databases that use archival logging) can be backed up at any time, regardless of whether the database is offline or online. However, before a database (recoverable or nonrecoverable) can be restored, it must first be taken offline. (Individual table spaces can be both backed up and restored while a database remains online.)

When an online back up operation is performed, archival logging ensures that all changes made while the back up image is being made are captured and can be re-created with a roll-forward recovery operation. Additionally, online back up operations can be performed against individual table spaces as well as entire databases. And, unlike when full database version recovery operations are performed, table space version recovery operations and table space roll-forward recovery operations can be performed while a database remains online—provided the table space that contains the system catalog is not the table space being recovered. While an online table space back up operation is performed, the table space being backed up remains available for use, and all modifications to the data stored in that table space are recorded in the transaction log files. However, when an online restore or online roll-forward recovery operation is performed against a table space, the table space itself is taken offline and is not available for use until the restore/roll-forward recovery operation is complete.

Incremental and delta back up and recovery

As the size of a database grows, the time and hardware needed to back up and recover the databases also grows substantially. Thus, creating full database and table space back up images is not always the best approach when dealing with large databases because the storage requirements for multiple copies of such back up images can be enormous. A better alternative is to create a full back up image periodically and one or more incremental back up images on a more frequent basis. An incremental back up is a back up image that
contains only pages that have been updated since the previous back
up image was made. Along with updated data and index pages, each
incremental back up image also contains all of the initial database
metadata (such as database configuration, table space definitions,
recovery history file, etc.) that is normally found in a full database
back up image.

Two types of incremental back up images can be produced:
incremental and delta. An incremental back up image is a copy of all
database data that has changed since the most recent, successful, full
back up image was created. An incremental back up image is also
known as a cumulative back up image because the last incremental
back up image in a series of incremental back up images made over a
period of time will contain the contents of all of the previous
incremental back up images. The predecessor of an incremental back
up image is always the most recent successful full back up image of
the same object.

A delta back up image, on the other hand, is a copy of all database
data that has changed since the last successful back up (full,
incremental, or delta) of the database or table space in question. For
this reason, a delta back up image is also known as a differential, or
noncumulative, back up image. The predecessor of a delta back up
image is the most recent successful back up image that contains a
copy of each of the objects found in the delta back up image.

The one thing that incremental and delta back up images have in
common is that before either type of back up image can be created; a
full back up image must already exist. Where they differ can be seen
both in their creation (usually, delta back up images are smaller and
can be created faster than incremental back up images) and in how
they are used for recovery. When incremental back up images are
taken, database recovery involves restoring the database using the
most recent full back up image available and applying the most
recent incremental back up image produced. On the other hand,
when delta back up images are taken, database recovery involves
restoring the database using the most recent full back up image
available and applying each delta back up image produced since the
full back up image used was made, in the order in which they were
created.
Performing a crash recovery operation

Earlier, we saw that whenever transaction processing is interrupted by an unexpected event (such as a power failure), the database with which the transaction was interacting at the time is placed in an inconsistent state. Such a database will remain in an inconsistent state and will be unusable until a crash recovery operation returns it to some point of consistency. (An inconsistent database will notify users and applications that it is unusable via a return code and error message that is generated each time an attempt to activate it or establish a connection to it is made.)

So just how is a crash recovery operation initiated? One way is by executing the `RESTART DATABASE` command from the DB2 Command Line Processor (CLP). The basic syntax for this command is:

```
RESTART [DATABASE | DB] [DatabaseAlias]
<User [UserName] <USING [Password]>>
<DROP PENDING TABLESPACES ([TS_Name], ... )>
<WRITE RESUME>
```

where:

- **DatabaseAlias** — Identifies the alias assigned to the database that is to be returned to a consistent and usable state.
- **UserName** — Identifies the name assigned to a specific user under whose authority the crash recovery operation is to be performed.
- **Password** — Identifies the password that corresponds to the name of the user under whom the crash recovery operation is to be performed.
- **TS_Name** — Identifies the name assigned to one or more table spaces that are to be disabled and placed in “Drop Pending” state during the crash recovery process.

If a problem occurs with a table space container during the restart process, the `DROP PENDING TABLESPACES ([TS_Name])` option can be used to place one or more table spaces in “Drop Pending” state. This allows the database to be successfully restarted, after which the offending table space(s) can be dropped and, if necessary, re-created. A list of troubled table space names can be found in the administration notification log if a database restart operation fails because of table space container problems.
Performing a crash recovery operation

Note: If there is only one system temporary table space in the database, and it is placed in “Drop Pending” state, a new system temporary table space must be created immediately following a successful database restart operation.

If all database I/O happened to be suspended at the time a crash occurred, the WRITE RESUME option of the RESTART command can be used to resume database I/O as part of the crash recovery process.

Thus, if you wanted to perform a crash recovery operation on an unusable database named SAMPLE, you could do so by executing a RESTART command that looks something like this:

RESTART DATABASE sample

On the other hand, if you wanted to perform a crash recovery operation on a database named SAMPLE and place a table space named TEMPSPACE1 in “Drop Pending” state, you could do so by executing a RESTART command that looks something like this:

RESTART DATABASE sample
DROP PENDING TABLESPACES (TEMPSPACE1)

You can also initiate a crash recovery operation for a particular database by selecting the Restart action from the Databases menu found in the Control Center. Figure 39 on page 150 shows the Control Center menu items that must be selected in order to perform a crash recovery operation on a DB2 database.
It is possible to configure a database in such a way that crash recovery will automatically be performed, if necessary, when an application or user attempts to establish a connection to it. This is done by assigning the value ON to the autorestart database configuration parameter. (DB2 checks the state of a database the first time an attempt to establish a connection to the database is made, and if it determines that the database is in an inconsistent state, it executes the RESTART command automatically if the autorestart database configuration parameter is set to ON — by default, when a database is created, the autorestart configuration parameter is set to ON.)
It is important to note that if a crash recovery operation is performed on a recoverable database (i.e., a database that has been configured to support forward recovery operations), and an error occurs during the recovery process that is attributable to an individual table space, that table space will be taken offline and will no longer be accessible until it is repaired. This has no effect on crash recovery itself, and upon completion of the crash recovery operation, all other table spaces in the database will be accessible and connections to the database can be established provided the table space that is taken offline is not the table space that contains the system catalogs. If the table space containing the system catalogs is taken offline, it must be repaired before any connections to the database will be permitted.

A word about soft checkpoints

It was mentioned earlier that crash recovery is performed by using information stored in the transaction log files to roll back all incomplete transactions found and complete any committed transactions that were still in memory (but had not yet been externalized to storage) when the transaction failure occurred. As you might imagine, if the transaction log files for a database are large, it could take quite a while to scan the entire log and check for corresponding rows in the database. However, it is usually not necessary to scan the entire log because records recorded at the beginning of a log file are usually associated with transactions that have been completed and have already been externalized to the database. Furthermore, if these records can be skipped, the amount of time required to recover a crashed database can be greatly reduced.

That’s where a mechanism known as the soft checkpoint comes in. DB2 uses a log control file to determine which records from a specific log file need to be applied to the database. This log control file is written to disk periodically, and the frequency at which this file is updated is determined by the value of the `softmax` database configuration parameter. Once the log control file is updated, the soft checkpoint information stored in it establishes where in a transaction log file crash recovery should begin; all records in a log file that precede the soft checkpoint are assumed to be associated with transactions that have already been written to the database and are ignored.
Back up and recovery

Although crash recovery can be used to resolve inconsistency problems that result from power interruptions or application failures, it cannot be used to handle problems that arise when the storage media being used to hold a database’s files becomes corrupted or fails. In order to handle these types of problems, some kind of backup (and recovery) program must be put in place.

A database recovery strategy should include a regular schedule for making database backup images and, in the case of partitioned database systems, include making backup images whenever the system is scaled (i.e., whenever database partition servers are added or dropped). Additionally, the strategy used should ensure that all information needed is available when database recovery is necessary, and it should include procedures for restoring command scripts, applications, user-defined functions (UDFs), stored procedure code in operating system libraries, and load copies as well as database data. To help with such a strategy, DB2 provides four utilities that are used to facilitate backing up and restoring a database:

- The Backup utility
- The Restore utility
- The Roll-forward utility
- The Recover utility

The DB2 Backup utility

The single most important item you can possess that will prevent catastrophic data losses in the event storage media becomes corrupted or fails is a database backup image. A database backup image is essentially a copy of an entire database that includes its metadata, its objects, its data, and optionally, its transaction logs. Once created, a backup image can be used at any time to return a database to the exact state it was in at the time the backup image was made (version recovery). A good database recovery strategy should ensure that backup images are created on a regular basis and that backup copies of critical data are retained in a secure location and on different storage media from that used to house the database. Depending on the logging method used (circular or archival), database backup images can be made when a database is offline or...
while other users and applications are connected to it. (In order to back up a database while it is online, archival logging must be enabled.)

A back up image of a DB2 database, or a table space within a DB2 database, can be created by executing the `BACKUP DATABASE` command. The basic syntax for this command is:

```
BACKUP [DATABASE | DB] [DatabaseAlias]
<User [UserName] <USING [Password]>>
<TABLESPACE ([TS_Name],...)
<ONLINE>
<INCREMENTAL <DELTA>>
<TO [Location] USE TSM <OPTIONS [TSMOptions]>>
<WITH [NumBuffers] BUFFERS>
<BUFFER [BufferSize]>
<PARALLELISM [ParallelNum]>
<UTIL_IMPACT_PRIORITY [Priority]>
<EXCLUDE LOGS | INCLUDE LOGS>
WITHOUT PROMPTING
```

where:

- `DatabaseAlias` — Identifies the alias assigned to the database for which a back up image is to be created.
- `UserName` — Identifies the name assigned to a specific user under whose authority the back up operation is to be performed.
- `Password` — Identifies the password that corresponds to the name of the user under whom the back up operation is to be performed.
- `TS_Name` — Identifies the name assigned to one or more specific table spaces for which back up images are to be created.
- `Location` — Identifies the directory or device where the back up image created is to be stored.
- `TSMOptions` — Identifies options that are to be used by Tivoli Storage Manager (TSM) during the back up operation.
- `NumBuffers` — Identifies the number of buffers that are to be used to perform the back up operation. (By default, two buffers are used if this option is not specified.)
- `BufferSize` — Identifies the size, in pages, that each buffer used to perform the back up operation will be. (By default, the size of each buffer used by the Backup utility is determined by the value of the `backbufsz` DB2 Database Manager configuration parameter.)
◆ **ParallelNum** — Identifies the number of table spaces that can be read in parallel during the back up operation.

◆ **Priority** — Indicates that the Backup utility is to be throttled such that it executes at a specific rate so that its effect on concurrent database activity can be controlled. This parameter can be assigned a numerical value within the range of 1 to 100, with 100 representing the highest priority and 1 representing the lowest.

If the **INCREMENTAL** option is specified, an incremental back up image will be produced—an incremental back up image is a copy of all data that has changed since the last successful, full back up image was produced. If the **INCREMENTAL DELTA** option is specified, a delta back up image will be produced—a delta back up image is a copy of all data that has changed since the last successful back up image of any type (full, incremental, or delta) was produced.

Thus, if you wanted to create a full back up image of a database named **SAMPLE** that is currently offline and store the image created in a directory named **BACKUPS** on logical disk drive **E:**, you could do so by executing a **BACKUP DATABASE** command that looks something like this:

```
BACKUP DATABASE sample
USER db2admin USING ibmdb2
TO E:\backups
```

On the other hand, if you wanted to create an incremental back up image of a table space named **TBSP1** and store the image created in a directory named **BACKUPS** on logical disk drive **E:** while the database it is associated with (named **SAMPLE**) remains online, you could do so by executing a **BACKUP DATABASE** command that looks something like this:

```
BACKUP DATABASE sample
USER db2admin USING ibmdb2
TABLESPACE (tbsp1) ONLINE INCREMENTAL TO E:\backups
```

Keep in mind that table space back up images can be created only if archival logging is being used; if circular logging is used instead, table space back up operations are not supported.

You can also create a back up image of a database or one or more table spaces using the Backup Wizard, which can be activated by selecting the Backup action from the Databases menu found in the Control Center. **Figure 40 on page 155** shows the Control Center.
menu items that must be selected to activate the Backup Wizard; Figure 41 on page 156 shows how the first page of the Backup Wizard might look immediately after it is activated.
Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

Figure 41  The first page of the Backup Wizard

Note: Only users with System Administrator (SYSADM) authority, System Control (SYSCTRL) authority, or System Maintenance (SYSSMAINT) authority are allowed to back up a database or its table spaces.

A word about back up image naming conventions

When a backup image is saved to disk, it is stored in a single flat file that has been assigned a unique name using the following naming convention:

```
DB_ALIAS.TYPE.INSTANCE.NODE####.CATN####.TIMESTAMP.SEQ_N
```

ICO IMG-0000000
The elements used in this naming convention are described in Table 10.

### Table 10 DB2 back up image file name components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DB_ALIAS</strong></td>
<td>A 1- to 8-character database alias that identifies the database to which the back up image belongs. (If no database alias exists, the database name is used instead.)</td>
</tr>
</tbody>
</table>
| **TYPE** | A number that identifies the back up image type, where:  
  - 0 represents a full, database-level back up image  
  - 3 represents a table space-level back up image  
  - 4 represents a back up image that was generated by the `LOAD...COPY TO` command |
| **INSTANCE** | A 1- to 8-character name of the current instance. (This value is obtained from the `DB2INSTANCE` environment variable.) |
| **NODE####** | The database partition number. In single partition database environments, this is always `NODE0000`. In partitioned database environments, it is `NODExxxx`, where `xxxx` is the number that has been assigned to the database partition in the `db2nodes.cfg` file. |
| **CAT####** | The database partition number of the catalog partition for the database. In single partition database environments, this is always `CATN0000`. In partitioned database environments, it is `CATNxxxx`, where `xxxx` is the number that has been assigned to the database partition in the `db2nodes.cfg` file that contains the system catalog for the database. |
| **TIMESTAMP** | A 14-character representation of the date and time that the back up image was created. This date/time representation is in the form `yyyyymmddhhnnss`, where:  
  - `yyyy` represents the year (1995 to 9999)  
  - `mm` represents the month (01 to 12)  
  - `dd` represents the day of the month (01 to 31)  
  - `hh` represents the hour (00 to 23)  
  - `nn` represents the minutes (00 to 59)  
  - `ss` represents the seconds (00 to 59) |
| **SEQ_NUM** | A 3-digit sequence number that is used as a file extension. This number differentiates multiple targets for the same back up image. |

Thus, a full database-level back up image for a single-partition database named `SAMPLE` that was created at 7:45 AM on March 26, 2008 would be assigned a name that looks something like this:

```
SAMPLE.0.DB2.NODE0000.CATN0000.20080326074544.001
```

When a back up image is written to tape, file names are not created but the information shown in Table 10 on page 157 is stored in the back up header for verification purposes.
The DB2 Restore utility

Earlier, we saw that version recovery is the process that returns a database to the state it was in at the time a backup image was made. This means that in order to perform a version recovery operation, at least one backup image must exist and be available. So just how is a version recovery operation initiated? The most common way is by executing the `RESTORE DATABASE` command. The basic syntax for this command is:

```
RESTORE [DATABASE | DB] [DatabaseAlias]
<USER [UserName] <USING [Password]>>
[TABLESPACE ([TS_Name],...) <ONLINE> |
 HISTORY FILE <ONLINE>> |
 COMPRESSION LIBRARY <ONLINE>> |
 LOGS <ONLINE>>]
<INCREMENTAL <AUTO | AUTOMATIC | ABORT>>
<FROM [SourceLocation] | USE TSM <OPTIONS [TSMOptions]]>
<TAKEN AT [Timestamp]>
<TO [TargetLocation]>
<DBPATH ON [TargetPath]>
<INFO [TargetAlias] > <LOGTARGET [LogsLocation]>
<NEWLOGPATH [LogsLocation]>
<WITH [NumBuffers] BUFFERS>
<BUFFER [BufferSize]>
<REPLACE HISTORY FILE>
<REPLACE EXISTING>
<REDIRECT <GENERATE SCRIPT [ScriptFile]>>
<PARALLELISM [ParallelNum]>
<WITHOUT ROLLING FORWARD>
<WITHOUT PROMPTING>
```

or

```
RESTORE [DATABASE | DB] [DatabaseAlias]
<USER [UserName] <USING [Password]>>
<REBUILD WITH [TABLESPACE ([TS_Name],...) |
 [ALL TABLESPACES IN [DATABASE | IMAGE]]
 <EXCEPT TABLESPACE ([TS_Name],...)>>
<INCREMENTAL <AUTO | AUTOMATIC | ABORT>>
<FROM [SourceLocation] | USE TSM <OPTIONS [TSMOptions]]>
<TAKEN AT [Timestamp]>
<TO [TargetLocation]>
<DBPATH ON [TargetPath]>
<INFO [TargetAlias] > <LOGTARGET [LogsLocation]>
<NEWLOGPATH [LogsLocation]>
<WITH [NumBuffers] BUFFERS>
<BUFFER [BufferSize]>
<REPLACE HISTORY FILE>
<REPLACE EXISTING>
```
<REDIRECT <GENERATE SCRIPT [ScriptFile]>>
<PARALLELISM [ParallelNum]>
<WITHOUT ROLLING FORWARD>
<WITHOUT PROMPTING>

or

RESTORE [DATABASE | DB] [DatabaseName]
[CONTINUE | ABORT]

where:

- **DatabaseAlias** — Identifies the alias assigned to the database associated with the backup image that is to be used to perform a version recovery operation.
- **UserName** — Identifies the name assigned to a specific user under whom the version recovery operation is to be performed.
- **Password** — Identifies the password that corresponds to the name of the user under whom the version recovery operation is to be performed.
- **TS_Name** — Identifies the name assigned to one or more specific table spaces that are to be restored from a backup image. (If the table space name has changed since the backup image was made, the new name should be specified.)
- **SourceLocation** — Identifies the directory or device where the backup image to be used for version recovery is stored.
- **TSMOptions** — Identifies options that are to be used by Tivoli Storage Manager (TSM) during the version recovery operation.
- **Timestamp** — Identifies a timestamp that is to be used as search criterion when looking for a particular backup image to use for version recovery. (If no timestamp is specified, it is assumed that there is only one backup image stored at the source location specified.)
- **TargetLocation** — Identifies the directory where the storage containers for the database that will be created are to be stored—if the backup image is to be used to create a new database and automatic storage is used.
- **TargetPath** — Identifies the directory where the metadata for the database that will be created is to be stored—if the backup image is to be used to create a new database and automatic storage is used.
◆ **TargetAlias** — Identifies the alias to be assigned to the new database to be created.

◆ **LogsLocation** — Identifies the directory or device where log files for the new database are to be stored.

◆ **NumBuffers** — Identifies the number of buffers that are to be used to perform the version recovery operation. (By default, two buffers are used if this option is not specified.)

◆ **BufferSize** — Identifies the size, in pages, that each buffer used to perform the back up operation will be. (By default, the size of each buffer used by the **RESTORE** utility is determined by the value of the *restbufsz* DB2 Database Manager configuration parameter.)

◆ **ScriptFile** — Identifies the name of a file to which all commands needed to perform a redirected restore operation are to be written.

◆ **ParallelNum** — Identifies the number of table spaces that can be read in parallel during the version recovery operation.

Thus, if you wanted to restore a database named **SAMPLE** (which already exists), using a full back up image stored in a directory named **BACKUPS** on logical disk drive **E:**, you could do so by executing a **RESTORE DATABASE** command that looks something like this:

```
RESTORE DATABASE sample
USER db2admin USING ibmdb2
FROM E:\backups
REPLACE EXISTING
WITHOUT PROMPTING
```

On the other hand, if you wanted to restore a table space named **TBSP1** from an incremental back up image stored in a directory named **BACKUPS** on logical disk drive **E:** while the database it is associated with (named **SAMPLE**) remains online, you could do so by executing a **RESTORE DATABASE** command that looks something like this:

```
RESTORE DATABASE sample
USER db2admin USING ibmdb2
TABLESPACE (tbsp1) ONLINE
INCREMENTAL
FROM E:\backups
```
Each full database back up image contains, among other things, a copy of the database's recovery history file. However, when an existing database is restored from a full database back up image, the existing recovery history file is not overwritten. But what if the recovery history file for the database happens to be corrupted? Can the recovery history file be restored as well given that a copy exists in the database back up image? The answer is yes. A special form of the RESTORE DATABASE command can be used to restore just the recovery history file from a database back up image. Such a RESTORE DATABASE command looks something like this:

```
RESTORE DATABASE sample
HISTORY FILE
FROM E:\backups
```

It is also possible to create an entirely new database from a full database back up image, effectively cloning an existing database. Therefore, you could create a new database named `SAMPLE_2` that is an exact duplicate of a database named `SAMPLE`, using a back up image stored in a directory named `BACKUPS` on logical disk drive `E:` by executing a RESTORE DATABASE command that looks something like this:

```
RESTORE DATABASE sample
USER db2admin USING ibmdb2
FROM E:\backups
INTO sample_2
```

It is important to note that if a back up image is used to create a new database, the recovery history file stored in the back up image will become the recovery history file for the new database.

You can also perform any of the restore/recovery operations just described (along with many others) using the Restore Data Wizard, which can be activated by selecting the Restore action from the Databases menu found in the Control Center. Figure 42 on page 162 shows the Control Center menu items that must be selected to activate the Restore Data Wizard; Figure 43 on page 163 shows how the first page of the Restore Data Wizard might look immediately after it is activated.
Figure 42  Invoking the Restore Data Wizard from the Control Center
Figure 43  The first page of the Restore Database Wizard

*Note:* Only users with System Administrator (SYSADM) authority, System Control (SYSCtrl) authority, or System Maintenance (SYSMaint) authority are allowed to restore a database or any of its table spaces from a back up image; only users with SYSADM authority or SYSCtrl authority are allowed to create a new database from a back up image.

**The DB2 Roll-forward utility**

When a back up image is used to restore a damaged or corrupted database, the database can only be returned to the state it was in at the time the back up image was made. Therefore, all changes that were made to the database after the back up image was created will be lost when a version recovery operation is performed. To return a database to the state it was in at any given point in time, roll-forward
recovery must be used instead. And in order to perform a roll-forward recovery operation, the database must be recoverable (that is, the database must be configured to use archival logging), you must have a full back up image of the database available, and you must have access to all archived log files that have been created since the last back up image (full, incremental, or delta) was made.

Roll-forward recovery starts out as a version recovery operation. However, where a version recovery operation will leave a nonrecoverable database in a “Normal” state, the same operation will leave a recoverable database in “Roll-forward pending” state (unless the WITHOUT ROLLING FORWARD option was specified with the RESTORE DATABASE command that was used to recover the database). At that point, either the database can be taken out of “Roll-forward pending” state (in which case all changes made to the database since the back up image used for version recovery was made will be lost), or information stored in the database’s transaction log files can be replayed to return the database to the state it was in at any given point in time.

The process of replaying transactions stored in archived log files is known as “rolling the database forward,” and one way to roll a database forward is by executing the ROLLFORWARD DATABASE command. The basic syntax for this command is:

ROLLFORWARD [DATABASE | DB] [DatabaseAlias]
<USER [UserName] <USING [Password]>>
<TO [PointInTime] <USING [UTC | LOCAL] TIME>
   <AND [COMPLETE | STOP]>
   END OF BACKUP <AND [COMPLETE | STOP]>
   END OF LOGS <AND [COMPLETE | STOP]>
   COMPLETE
   STOP
   CANCEL
   QUERY STATUS <USING [UTC | LOCAL] TIME>>
<TABLESPACE ONLINE |
   TABLESPACE <([TS_Name], ...) <ONLINE>>
<OVERFLOW LOG PATH ([LogDirectory], ...)>
<RECOVER DROPPED TABLE [TableID] TO [Location]>

where:
- DatabaseAlias — Identifies the alias assigned to the database that is to be rolled forward.
- UserName — Identifies the name assigned to a specific user under whom the roll-forward operation is to be performed.
- **Password** — Identifies the password that corresponds to the name of the user under whom the roll-forward operation is to be performed.
- **PointInTime** — Identifies a specific point in time, identified by a timestamp value in the form `yyyy-mm-dd-hh.mm.ss.nnnnnn` (year, month, day, hour, minutes, seconds, microseconds), to which that the database is to be rolled forward. (Only transactions that took place before and up to the date and time specified will be reapplied to the database.)
- **TS_Name** — Identifies the name assigned to one or more specific table spaces that are to be rolled forward. (If the table space name has changed since the back up image used to restore the database was made, the new name must be specified.)
- **LogDirectory** — Identifies the directory that contains offline archived log files that are to be used to perform the roll-forward recovery operation.
- **TableID** — Identifies a specific table (by ID) that was dropped earlier that is to be restored as part of the roll-forward operation. (The table ID can be obtained by examining the database’s recovery history file.)
- **Location** — Identifies the directory to which files containing dropped table data are to be written when the table is restored as part of the roll-forward operation.

If the `AND COMPLETE, AND STOP, COMPLETE, or STOP` option is specified, the database will be returned to “Normal” state when the roll-forward operation has completed. Otherwise, the database will remain in “Roll-forward pending state.” (When a recoverable database is restored from a back up image, it is automatically placed in “Roll-forward pending” state unless the `WITHOUT ROLLING FORWARD` option is used with the `RESTORE DATABASE` command; while a database is in “Roll-forward pending” state, it cannot be accessed by users and applications.)

If the `QUERY STATUS` option is specified, a list of the log files that have been used to perform the roll-forward recovery operation, along with the next archive file required, and the time stamp of the last committed transaction since roll-forward processing began is returned.
Thus, if you wanted to perform a roll-forward recovery operation on a database named `SAMPLE` that was just restored from a backup image, you could do so by executing a `ROLLFORWARD DATABASE` command that looks something like this:

```
ROLLFORWARD DATABASE sample TO END OF LOGS AND STOP
```

On the other hand, if you wanted to perform a roll-forward recovery operation on a table space named `DATA_TBS` in a database named `SAMPLE` by reapplying transactions that were performed against the table space on or before April 1, 2007 (04/01/2007), you could do so by executing a `ROLLFORWARD DATABASE` command that looks something like this:

```
ROLLFORWARD DATABASE sample TO 2007-04-01-00.00.00.0000 AND STOP TABLESPACE (data_tbsp)
```

It is important to note that the time value specified is interpreted as a Coordinated Universal Time (UTC) — otherwise known as Greenwich Mean Time (GMT) — value. If a `ROLLFORWARD DATABASE` command that looks something like the following had been executed instead, the time value specified would have been interpreted as a local time value:

```
ROLLFORWARD DATABASE sample TO 2007-04-01-00.00.00.0000 USING LOCAL TIME AND STOP TABLESPACE (data_tbsp)
```

Note: When rolling a table space forward to a specific point in time, the time specified must be greater than the minimum recovery time recorded for the table space. This time can be obtained by executing the command `LIST TABLESPACES SHOW DETAIL`. Among other things, this command returns the earliest point in time to which each table space can be rolled forward. The minimum recovery time is updated when data definition language (DDL) statements are run against a table space or against tables stored in a table space. A table space must be rolled forward to at least the minimum recovery time so that it becomes synchronized with the information in the system catalog tables. If recovering more than one table space, each table space must be rolled forward to at least the highest minimum recovery time of all the table spaces being recovered.

If you want to roll a table space forward to a specific point in time, and a table in the table space participates in a referential integrity constraint with another table that resides in another table space, you should roll both table spaces forward simultaneously to the same point in time. If you do not, the child table in the referential integrity
relationship will be placed in “Set integrity pending” state at the end of the roll-forward recovery operation, and constraint checking will have to be performed on the table before it can be used.

You can also initiate a roll-forward recovery operation using the Roll-forward Wizard, which can be activated by selecting the Roll-forward action from the Databases menu found in the Control Center. Figure 44 shows the Control Center menu items that must be selected to activate the Roll-forward Wizard; Figure 45 on page 168 shows how the first page of the Roll-forward Wizard might look immediately after it is activated.

![Figure 44 Invoking the Roll-forward Wizard from the Control Center](image-url)
Because a roll-forward recovery operation is typically performed immediately after a database is restored from a backup image, a roll-forward recovery operation can also be initiated by providing the appropriate information on the Roll-forward page of the Restore Data Wizard. Figure 46 on page 169 shows how the Roll-forward page of the Restore Data Wizard normally looks.
A word about the recovery history file

In Chapter 1, “Introduction to DB2 for Linux, UNIX, and Windows,” we saw that a special file, known as the recovery history file, is created as part of the database creation process. This file is used to log historical information about specific actions that are performed against the database with which the file is associated. Specifically, records are written to the recovery history file whenever any of the following actions are performed:

- A table space is created.
A table space is altered.
A table space is quiesced.
A table space is renamed.
A table space is dropped.
A table is loaded.
A table is dropped (when dropped table recovery is enabled).
A table is reorganized, using the REORG utility.
Statistics for a table are updated, using the RUNSTATS utility.
A table is loaded, using the Load utility.
The database or one of its table spaces is backed up.
The database or one of its table spaces is restored.
The database or one of its table spaces is rolled forward.
The database is recovered.
On-demand log archiving is invoked.
A new log file is written to (when using recoverable logging).
A log file is archived (when using recoverable logging).
The database is automatically rebuilt and more than one image is restored.

In addition to identifying the event that was performed, each entry in the recovery history file identifies the date and time the event took place, how the event took place, and the table spaces and tables that were affected. If the action was a back up operation, the location where the back up image produced was stored, along with information on how to retrieve it; because the recovery history file contains image location information for each back up image available, it acts as a tracking and verification mechanism during version recovery operations. In fact, the information stored in the recovery history file is what drives automatic recovery — automatic recovery is initiated by executing either the RESTORE DATABASE or the RECOVER DATABASE command with the AUTOMATIC option specified.

Because the recovery history file sits quietly in the background and DB2 is responsible for managing its contents, a database administrator rarely has to interact with it. However, two commands — LIST HISTORY and PRUNE HISTORY — provide a way to both view the contents of a database’s recovery history file and remove
one or more entries stored in it. Additionally, if the recovery history file for a database becomes corrupt, it is possible to restore just the recovery history file from a database backup image.

## The DB2 Recover utility

Earlier, we saw how the **RESTORE DATABASE** command can be used to return a database to the state it was in at the time a backup image was made and how the **ROLLFORWARD DATABASE** command can be used to replay information recorded in a database’s transaction log files to return a database to the state it was in at a specific point in time. If you are restoring the database from a full database backup image, this process is pretty straightforward. However, if you have a full database backup image and several incremental backup images, delta backup images, or table space backup images, the process can be a little complicated. That’s where DB2’s Recover utility comes in.

Introduced in DB2 9, the Recover utility performs the necessary restore and roll-forward operations to recover a database to a specific point in time, based on information found in the recovery history file. The Recovery utility is invoked by executing the **RECOVER DATABASE** command. The basic syntax for this command is:

```sql
RECOVER [DATABASE | DB] [DatabaseAlias]
  <TO [PointInTime] <USING [UTC | LOCAL] TIME>
  <ON ALL DBPARTITIONNUMS> | END OF LOGS <ON ALL DBPARTITIONNUMS>
  ON DBPARTITIONNUM<S> [PartitionNum],....>
  <USER [UserName] <USING [Password]>>
  <USING HISTORY FILE ([HistoryFile])>
  <OVERFLOW LOG PATH ([LogDirectory],....)>
  <COMPLIB [CompLibrary] > <COMPROPTS [CompOptions]>
  <RESTART>
```

where:
- **DatabaseAlias** — Identifies the alias assigned to the database associated with the backup image that is to be used to perform a version recovery operation.
- **PointInTime** — Identifies a specific point in time, identified by a timestamp value in the form `yyyy-mm-dd-hh.mm.ss.nnnnnn` (year, month, day, hour, minutes, seconds, microseconds), to which the database is to be rolled forward. (Only transactions that took place before and up to the date and time specified will be reapplied to the database.)
PartitionNum — Identifies one or more database partitions (as specified in the `db2nodes.cfg` file) that roll-forward recovery is to be performed on.

UserName — Identifies the name assigned to a specific user under whom the recovery operation is to be performed.

Password — Identifies the password that corresponds to the name of the user under whom the recovery operation is to be performed.

HistoryFile — Identifies the name assigned to the recovery history log file that is to be used by the Recovery utility.

LogDirectory — Identifies the directory that contains offline archived log files that are to be used to perform the roll-forward portion of the recovery operation.

CompLibrary — Identifies the name of the library that is to be used to decompress the data stored in the backup image. (This parameter is only used if the backup image was compressed.)

CompOptions — Describes a block of binary data that is to be passed to the initialization routine in the decompression library used.

Thus, if you wanted to perform a full recovery operation on a database named `SAMPLE` (which already exists) using information stored in the recovery history file, you could do so by executing a `RECOVER DATABASE` command that looks something like this:

```
RECOVER DATABASE sample TO END OF LOGS
```

On the other hand, if you wanted to restore a database named `SAMPLE` and roll it forward to an extremely old point in time that is no longer contained in the current recovery history file, you could do so by executing a `RECOVER DATABASE` command that looks something like this (assuming you have a copy of an older recovery history file available):

```
RECOVER DATABASE sample
TO 2005-01-31-04.00.00
USING HISTORY FILE (/home/user/old2005files/db2rhist.asc)
```
Rebuilding invalid indexes

So far we have looked at ways that data can be recovered in the event the storage media being used to hold a database’s files becomes corrupted or fails. But what if only indexes are damaged, and a database’s data is unaffected (which could be the case if data and indexes are stored in separate table spaces and only the physical device where index data is stored fails)? In this case, the affected indexes are invalidated and can be recovered by being re-created once the faulty media has been replaced.

Whenever the DB2 Database Manager detects that an index is no longer valid, it automatically attempts to rebuild it. However, the point in time at which the DB2 Database Manager attempts to rebuild an invalid index is controlled by the `indexrec` parameter of the database or the DB2 Database Manager configuration file. There are three possible settings for this parameter:

- **SYSTEM.** Invalid indexes are to be rebuilt at the time specified in the `indexrec` parameter of the DB2 Database Manager configuration file. (This setting is valid only for database configuration files.)

- **RESTART.** Invalid indexes are to be rebuilt, either explicitly or implicitly, when the database is restarted (i.e., when crash recovery is performed on the database).

- **ACCESS.** Invalid indexes are to be rebuilt the first time they are accessed after they have been marked as being invalid.

So when is the best time to rebuild invalid indexes? If the time needed to perform a crash recovery operation on a database is not a concern, it is better to let the DB2 Database Manager rebuild invalid indexes while it is in the process of returning the database to a consistent state; the time needed to restart a database will be longer because of the index recreation process, but once the database has been restored, query processing will not be impacted. On the other hand, if indexes are rebuilt as they are accessed, crash recovery will be performed faster, but users may experience a decrease in performance—queries against tables that contain associated invalid indexes will have to wait for the invalid indexes to be rebuilt before they can be processed. Furthermore, unexpected locks may be acquired and held long after an invalid index has been re-created, especially if the transaction that caused the index recreation to occur is not committed (or rolled back) for some time.
Although the `indexrec` parameter of the database or the DB2 Database Manager configuration file can be used to control when indexes are rebuilt as part of a crash recovery operation, it has no affect on how indexes are rebuilt during roll-forward recovery operations. To control that behavior, you must assign the appropriate value to the `logindexbuild` database configuration parameter. There are two possible settings for this parameter:

- **ON.** Index creation, re-creation, and reorganization operations are to be recorded in the database's transaction log files so that indexes can be reconstructed during roll-forward recovery operations or high availability disaster recovery (HADR) log replay operations.

- **OFF.** Index creation, re-creation, and reorganization operations will not be recorded in the database's transaction log files.

If the `LOG INDEX BUILD` table attribute is set to its default value of `NULL`, DB2 will use the value specified for the `logindexbuild` database configuration parameter. If the `LOG INDEX BUILD` table attribute is set to `ON` or `OFF`, the value specified for the `logindexbuild` database configuration parameter will be ignored.

### Backing up a database with split mirroring

It was mentioned earlier that as databases increase in size and as heavy usage demands require databases to be available 24 hours a day, 7 days a week, the time and hardware needed to back up and restore a database can increase substantially. Backing up an entire database or several table spaces of a large database can put a strain on system resources, require a considerable amount of additional storage space (to hold the back up images), and reduce the availability of the database system (particularly if the system has to be taken offline in order to be backed up). Therefore, a popular alternative to creating and maintaining back up images of high-availability databases is to use what is known as a split mirror.

A split mirror is an “instantaneous” copy of a database that is made by mirroring the disk or disks that contain the database's data and splitting the mirror when a back up copy of the database is required. Mirroring is the process of writing all database data to two separate disks (or disk subsystems) simultaneously; one disk/subsystem holds the database data while the other holds an exact copy (known as a mirror) of the primary disk/subsystem being used. Splitting a
mirror simply involves separating the primary and secondary copies of the database from each other. Split mirroring provides the following advantages:

- The overhead required to create back up images of the database is eliminated.
- Entire systems can be cloned very quickly.
- It provides a fast implementation of idle standby failover.

To further enhance split mirroring, DB2 provides a way to temporarily suspend (and later resume) all database I/O so that a mirror can be split without having to take a database offline. The command that provides this functionality is the `SET WRITE` command, and the syntax for this command is:

```
SET WRITE [SUSPEND | RESUME] FOR [DATABASE | DB]
```

Therefore, if you wanted to temporarily suspend all I/O for a database, you would do so by establishing a connection to that database and executing a `SET WRITE` command that looks like this:

```
SET WRITE SUSPEND FOR DATABASE
```

When executed, the `SET WRITE SUSPEND FOR DATABASE` command causes DB2 to suspend all write operations to table space containers and log files that are associated with the current database. (The suspension of writes to table spaces and log files is intended to prevent partial page writes from occurring until the suspension is removed.) All operations, apart from online back up and restore operations, will function normally while database writes are suspended. That's because read-only transactions are not suspended and are able to continue working with a write-suspended database; applications can continue to perform insert, update, and delete operations with data that has been cached in the database's buffer pool(s), but new pages cannot be written to disk. (Data can still be read from disk; only write operations are suspended.) Additionally, new database connections can be established to a write-suspended database provided the system catalog pages required to authenticate the connections already reside in a buffer pool.

I/O for a write-suspended database can be resumed at any time by executing a `SET WRITE` command that looks like this:

```
SET WRITE RESUME FOR DATABASE
```
When executed, the **SET WRITE RESUME FOR DATABASE** command causes DB2 to lift all write suspensions and to allow write operations to table space containers and log files that are associated with the current database to continue.

**IMPORTANT**

The **SET WRITE RESUME FOR DATABASE** command must be executed from the same connection from which the **SET WRITE SUSPEND FOR DATABASE** command was issued.

---

**Initializing a split mirror with db2inidb**

Before a split mirror copy of a DB2 database can be used, it must first be initialized; a split mirror database copy is initialized by executing the system command **db2inidb**. The syntax for this command is:

```
db2inidb [DatabaseAlias] AS [SNAPSHOT | MIRROR | STANDBY] <RELOCATE USING [ConfigFile]>
```

where:

- **DatabaseAlias** — Identifies the alias assigned to the database that the split mirror copy to be initialized references.
- **ConfigFile** — Identifies that the database files contained in the split mirror copy are to be relocated according to information stored in the configuration file specified.

As you can see, a split mirror database copy can be initialized in one of three ways:

- **SNAPSHOT**. The split mirror copy of the database will be initialized as a read/write clone of the primary database.
- **MIRROR**. The split mirror copy of the database will be initialized as a back up image that can be used to restore the existing primary database.
- **STANDBY**. The split mirror copy of the database will be initialized and placed in roll-forward pending state so that it can be continuously synchronized with the primary database. (New logs from the primary database can be retrieved and applied to the copy of the database at any time.) The standby copy of the database can then be used in place of the primary database if, for some reason, the primary database goes down.
Note: If you execute the `db2inidb` command with the `STANDBY` option specified, you can invoke the DB2 Backup utility immediately afterwards provided there are no SMS table spaces being used for either the system catalogs OR any user data. (SMS temporary table spaces are tolerated.)

Thus, if you wanted to initialize a split mirror copy of a database named `SAMPLE` and make it a back up image that can be used to restore the primary database, you could do so by executing a `db2inidb` command that looks like this:

```
db2inidb sample AS MIRROR
```
The EMC TimeFinder family of local replication products enables users to nondisruptively create and manage point-in-time copies of data to allow operational processes such as backup image generation, reporting, and application testing to be performed independently of the source data to maximize service levels without impacting performance or availability.

This chapter is designed to introduce you to the base replication products and component options that make up the EMC TimeFinder family. Topics covered include:

- Introduction ................................................................. 180
- Symmetrix device types used in TimeFinder operations........ 183
- Performing TimeFinder/Clone operations ......................... 188
- Performing TimeFinder/Snap operations .......................... 190
- Performing TimeFinder/Mirror operations ....................... 193
Introduction

The EMC TimeFinder family of software is the most powerful suite of local storage replication solutions available. Fully leveraging the industry-leading high-end Symmetrix hardware architecture, it offers unmatched deployment flexibility and massive scalability to deliver a wide range of in-the-system data copying capabilities to meet mixed service level requirements without operational impact.

The TimeFinder family can be used in environments configured with EMC Symmetrix VMAX arrays and Symmetrix DMX™ arrays. Symmetrix VMAX arrays require EMC Enginuity Operating Environment for Symmetrix systems release 5874 or later. Symmetrix DMX arrays require Enginuity 5773 and earlier.

Currently, the TimeFinder family consists of two separate and distinct software products along with several additional component options. The TimeFinder base replication products consist of:

- **TimeFinder/Clone** enables users to make 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, 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 TimeFinder BCV device.

- **TimeFinder/Snap** enables users to configure special devices in the Symmetrix array called *virtual devices (VDEVs)* and *save 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 TimeFinder 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.

TimeFinder component options include:

- **TimeFinder/Mirror** enables users to configure special devices, called *business continuance volumes (BCVs)*, to create a mirror image of Symmetrix standard devices. Using *BCV devices*, TimeFinder creates a point-in-time copy of data that...
EMC TimeFinder Family of Products

can be repurposed. TimeFinder/Mirror is a family component that works in Symmetrix environments running Enginuity 5773 and earlier. In environments running Enginuity 5874 and higher, all TimeFinder/Mirror scripts are executed in Clone Emulation mode.

IMPORTANT
Starting with Enginuity 5874, TimeFinder/Mirror functions are performed through TimeFinder/Clone software using a process called Clone Emulation. When running in Emulation Mode, TimeFinder/Clone transparently performs TimeFinder/Mirror commands and executes scripts that were written for Solutions Enabler up through version 6.5.2 running on Symmetrix arrays using Enginuity 5773 and earlier.

- **TimeFinder/Consistency Groups (TimeFinder/CG)** enables other TimeFinder family products to coordinate cross-volume and cross-system consistency to ensure application restartability. This option for Symmetrix arrays creates multivolume sets of point-in-time copies at the same instant, ensuring that as a set the copies are consistent and restartable, even when spread across multiple Symmetrix volumes and data is spread across multiple arrays, without quiescing or shutting down to ensure that all devices are copied at the same point in time.

- **TimeFinder/Exchange Integration Module (TimeFinder/EIM)** automates and simplifies the process of creating and managing TimeFinder replications of a Microsoft Windows Exchange Server environment.

- **TimeFinder/SQL Integration Module (TimeFinder/SIM)** automates (TimeFinder/SIM) simplifies the process of creating and managing TimeFinder replications of a Microsoft Windows SQL Server environment.

- **Duplicate TimeFinder/Snap** offers the capability to capture TimeFinder/Snap replicas from another TimeFinder/Snap point-in-time copy. This functionality is targeted mainly at SAP and database environments where copies of production environments are repurposed for testing, QA, or development. Work can proceed against an existing TimeFinder/Snap while duplicate copies can be created for additional downstream processes or checkpoint backups. With Enginuity 5875, snap
copies can be taken from another snap source, adding even more disk space savings and flexibility through this track sharing technology.

The SYMCLI TimeFinder component extends the basic SYMCLI command set to include TimeFinder or business continuance commands that perform control operations on device pairs within the TimeFinder environment. The commands that comprise the TimeFinder component technologies of the EMC Solutions Enabler are: `symclone, symsnap, symmir, symbcv`, and `symioctl`.

The TimeFinder/Clone command `symclone` is used to create a point-in-time copy of a source device on nonstandard device pairs (such as standard/standard and BCV/BCV).

The TimeFinder/Snap command `symsnap` is used to create virtual device copy sessions between a source device and multiple virtual target devices. These virtual devices only store pointers to changed data blocks from the source device, rather than a full copy of the data.

TimeFinder base component commands such as `symmir` and `symbcv` are used to perform a wide spectrum of monitor and control operations on Symmetrix standard/BCV device pairs within a TimeFinder environment. The `symioctl` command is used to send I/O control commands to a specified database server.
Symmetrix device types used in TimeFinder operations

Several different types of Symmetrix device are used in TimeFinder operations. The most common types of devices used include:

- Standard
- Business continuance volume (BCV)
- Clone
- Virtual (VDEV)
- SAVE (SAVEDEV)
- Meta
- DATA
- Thin

Standard device

A standard device is a Symmetrix storage device (disk) that is configured for normal Symmetrix operations under a desired protection method (SRDF, RAID S, RAID 1, RAID 5, RAID 6, and so on).

Standard devices can have any mirror structure (unprotected, one mirror, two mirrors), provided the number of mirrors does not exceed three. This constraint is in place because establishing a “BCV device” pair requires assigning the BCV device as the next available mirror of the standard device. When TimeFinder/Clone is in emulation mode, this constraint does not apply — the BCV device is not assigned as the next mirror.

Business continuance volume (BCV) device

A business continuance volume (BCV) device is a standard Symmetrix device that contains a copy of data from a standard Symmetrix device that is online for regular I/O operations from one or more of its hosts. The BCV is a full volume mirror or spare that has valid data after it has been fully synchronized with its paired source device. The BCV is accessible only when split from the source device that it is mirroring.
Uses for the BCV copies can include backup, restore, decision support, and applications testing. Each BCV device has its own host address, and is configured as a stand-alone Symmetrix device, with special attributes that allow it to be accessed independently of its associated standard device to support host applications and processes.

A business continuance sequence first involves establishing the BCV device as a mirror of a specific standard Symmetrix device. As a result, the BCV device becomes inaccessible via its original device address while it is in an established pair. Once the BCV device is synchronized, it may be separated (split) later from the standard device with which it is paired. Once split, the BCV device with the synchronized data becomes available for backup or other host processes through its original device address. Once host processing on the BCV device is complete, the BCV may again be mirrored to a standard Symmetrix device, which can either be the same device to which it was previously paired, or a different device.

**Clone device**

A clone device is a full-volume Symmetrix device used in TimeFinder/Clone operations to create a physical point-in-time Copy of a logical standard device on multiple target devices. The data is physically copied from the standard device, creating a physical backup copy, which can be used in case the STD becomes inaccessible.

TimeFinder/Clone copies are appropriate in situations where multiple copies of production data are needed for testing, backups, or report generation. Clone copies can also be used to reduce disk contention and improve data access speed by assigning users to copies of data rather than accessing the one production copy.

The source and target devices can be either standard devices or BCV devices as long as they are all of the same size and emulation type (FBA or CKD). Clone copies of striped or concatenated metadevices can also be created, but the source and target metadevices must be completely identical in stripe count, stripe size, and capacity. Once activated, the copy can be instantly accessed by a target’s host, even before the data is fully copied to the target device.
Virtual device (VDEV)

A virtual device (VDEV) is a Symmetrix host-accessible, logical-image device that contains track-level location information (pointers) indicating where the copy session data is located in the physical storage and is immediately accessible after activation.

Virtual devices are used in TimeFinder/Snap operations and consume minimal physical disk storage and contain only the address pointers to the data that is stored on the source device or in a pool of “SAVE devices.”

When a virtual copy session is activated, a point-in-time copy of the source device is immediately available to its host through the corresponding virtual device. Snapping data to a virtual device uses a copy-on-first-write technique. Upon a first write to the source device during the copy session, a preupdated image of the changed track is copied to a SAVE device. The track pointer on the virtual device will then be updated to point to the data on the SAVE device.

The attached host views the point-in-time copy through virtual device pointers to both the source device and SAVE device, for as long as the session remains active. If the copy session is terminated, the copy is lost and the space associated with the session is freed and returned to the SAVE device pool for future use. Optionally, before terminating a copy session, virtual devices can be incrementally or fully restored back to the original source, a BCV device split from its source, or another Symmetrix device.

SAVE device (SAVEDEV)

A SAVE device, or SAVEDEV, is a Symmetrix device that is not host-accessible and can only be accessed through the virtual devices that point to it. SAVE devices are configured to provide pooled physical storage to TimeFinder/Snap and SRDF/A operations:

- With TimeFinder/Snap, SAVE devices are organized into snap pools and are used to store pre-update images or changed tracks during a virtual copy session. Devices within a snap pool act as a group for storing data in striped form. Symmetrix supports the creation of multiple named snap pools, allowing symsnapshot commands to use a particular pool. When used in snap pools,
SAVE devices contain any original tracks that were changed as a result of a first copy on write to a source device or a new write to a virtual device during a virtual device copy session.

- With SRDF/A, SAVE devices are organized into Delta Set Extension pools and are used to extend the cache space available for SRDF/A cycles by offloading some or all of the cycle data from cache to preconfigured disk storage pools.

SAVE devices are assigned a Symmetrix device number and can be configured as unprotected, mirrored, or parity RAID. They cannot be part of a metavolume or grouped in device or composite groups.

---

**Metadevice**

A metadevice, or metavolume, consists of two or more individual devices that have been logically concatenated, creating larger storage devices that can be presented to the host as a single addressable device. The Metahead is the first device in the chain of metadevices or metamembers and is the initial target as it is a responsible gatekeeper for keeping incoming commands. The Metahead handles all command processing activities; when the Metahead is addressed with a command, Enginuity determines which metadevice is the target for the command execution.

Besides concatenation, metavolumes can be striped for better performance. A striped metadevice is one that places data on metamembers in user-defined stripes or chunks instead of filling an entire device first before addressing the next device. The striping of data across multiple drives effectively creates definable cylinder stripes.

---

**DATA device**

A DATA device is a specifically configured devices within the Symmetrix array that is not addressable to a host and is a container for the written-to blocks of thin devices. Any number of DATA devices may comprise a data device pool (also referred to as a thin pool). Blocks are allocated to the thin devices from the pool on a round robin basis. This allocation block size is 768 KB.
When DATA devices are added to a thin pool, the devices can be in an enabled or disabled state. To use a DATA device for thin extent allocation, the devices must be in the enabled state. To remove devices from the thin pool, the devices must be in a disabled state.

**Thin device**

A thin device is a host-accessible, logical Symmetrix device used in ways similar to the ways Symmetrix devices have traditionally been used. Like traditional devices, thin devices are provisioned by mapping them to a front-end Symmetrix director and port, and then LUN-masking them to the WWN of an HBA in the server requiring the storage.

Unlike traditional Symmetrix devices, thin devices have no storage directly associated with it — they do not require physical storage to be completely allocated at the time the device is created and presented to a host. They have preconfigured sizes and appear to the host to have that exact capacity. Device storage is allocated in chunks when a block is written to for the first time. Zeros are provided to the host for data that is read from chunks that have not yet been allocated. The physical storage that is used to supply disk space (storage capacity) to thin devices comes from a shared storage pool called a thin pool.

**IMPORTANT**

**BCV devices cannot be thin devices.**
Performing TimeFinder/Clone operations

Symmetrix TimeFinder/Clone operations are performed using the SYMCLI TimeFinder symclone command which creates clone copies of a source device on multiple target devices. The source and target devices can be either standard devices or BCV devices as long as they are all of the same size and emulation type (FBA or CKD). Clone copies of striped or concatenated metadevices can also be created, but the source and target metadevices must be completely identical in stripe count, stripe size, and capacity. Once activated, the copy can be instantly accessed by a target’s host, even before the data is fully copied to the target device.

There are several key advantages of using TimeFinder/Clone, such as the ability to perform precopy operations and its cache partitioning. TimeFinder/Clone copies are appropriate in situations where multiple copies of production data are needed for testing, backups, or report generation. Clone copies can also be used to reduce disk contention and improve data access speed by assigning users to copies of data rather than accessing the one production copy.

Depending on whether a device has associated BCVs, a single source device can have up to 16 clone copy sessions (15 copy sessions and one reserve copy session for restore operations). When using the -copy option, you can copy up to eight full data copies simultaneously, without disruption to database production activity.

Clone copy sessions

TimeFinder/Clone functionality is controlled via copy sessions, which pair source and target devices. Sessions are maintained on the Symmetrix system and can be queried to verify the current state of the device pairs. A copy session must first be created to define and set up the TimeFinder/Clone devices. The session is then activated, enabling the target device to be accessed by its host. When the information is no longer needed, the session can be terminated. TimeFinder/Clone operations are controlled from the host by using the symclone command to create, activate, restore, recreate, set mode, split, establish, and terminate copy sessions.

Figure 47 on page 189 illustrates a copy session where the controlling host creates a TimeFinder/Clone copy of standard device DEV001 on target device DEV005, using the symclone command.
Performing TimeFinder/Clone operations

EMC TimeFinder Family of Products

Figure 47 Creating a copy session using the symclone command

The `symclone` command is used to enable cloning operations. The cloning operation happens in two phases: creation and activation. The creation phase builds bitmaps of the source and target that are later used during the activation or copy phase. The creation of a symclone pairing does not start copying of the source volume to the target volume, unless the `-precopy` option is used.

For example, to create clone sessions on all the standards and BCVs in the device group MyDevGrp, use the following command:

```
symclone -g MyDevGrp create -noprompt
```

The activation of a clone enables the copying of the data. The data may start copying immediately if the `-copy` keyword is used. If the `-copy` keyword is not used, tracks are only copied when they are accessed from the target volume or when they are changed on the source volume.

Activation of the clone session established in the previous `create` command can be accomplished using the following command.

```
symclone -g MyDevGrp activate -noprompt
```
Performing TimeFinder/Snap operations

Symmetrix arrays provide another technique for creating copies of application data. The functionality, called TimeFinder/Snap, allows users to make pointer-based, space-saving copies of data simultaneously on multiple virtual (VDEV) target devices from a single source device. (The data is available for access instantly.)

TimeFinder/Snap allows data to be copied from a single source device to as many as 128 target devices (Enginuity 5772 and later). A source device can be either a Symmetrix standard device or a BCV device controlled by TimeFinder/Mirror, with the exception being a BCV working in clone emulation mode. The target device is a Symmetrix virtual device (VDEV), which consumes negligible physical storage through the use of pointers to track changed data.

A VDEV is a logical-image device that offers a space-saving way to create instant, point-in-time copies of volumes. Any updates to a source device after its activation with a VDEV, causes the pre-update image of the changed tracks to be copied to a SAVE device. The VDEV’s indirect pointer is then updated to point to the original track data on the SAVE device, preserving a point-in-time image of the volume. TimeFinder/Snap uses this copy-on-first-write technique to conserve disk space, since only changes to tracks on the source cause incremental storage to be consumed.

The `symsnap create` and `symsnap activate` commands are used to create a source/target snap pair.

Table 11 summarizes some of the differences between devices used in TimeFinder/Snap operations.

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual device</td>
<td>A logical-image device that saves disk space through the use of pointers to track data that is immediately accessible after activation. Snapping data to a virtual device uses a copy-on-first-write technique.</td>
</tr>
<tr>
<td>Save device</td>
<td>A device that is not host-accessible but accessed only through the virtual devices that point to it. Save devices provide a pool of physical space to store snap copy data to which virtual devices point.</td>
</tr>
<tr>
<td>BCV</td>
<td>A full volume mirror that has valid data after fully synchronizing with its source device. It is accessible only when split from the source device that it is mirroring.</td>
</tr>
</tbody>
</table>
TimeFinder/Snap operations can also be used to create copies of striped or concatenated metadevices provided the source and target virtual metadevices are completely identical in stripe count, stripe size, and capacity.

TimeFinder/Snap for virtual device copy operations provides multiple copies of production data for testing, backups or report generation. Such copies can also be used to reduce disk contention and to improve data access speed, by assigning users to copies of data rather than accessing the one production copy.

Virtual snap copies can also prove useful in situations where multiple backups or recovery copies of a source device are taken throughout the day, with only a small percentage of data actually changing on the device. As an alternative to using multiple BCVs to backup these devices, you can use virtual devices to reduce costs and save on storage space. Through the use of device pointers to the original data, virtual devices allow you to allocate space based on the expected amount of the changes to a device, rather than the size of the volume being backed up.

Snap copy sessions

Much like TimeFinder/Clone, TimeFinder/Snap functionality is managed via copy sessions, which pair source and target virtual devices. Sessions are maintained on the Symmetrix system and can be queried to verify the current state of the devices. A copy session must be created first—this defines the devices that will be used in the copy operation. On subsequent activation, the target virtual devices become accessible to a host. (Once a copy session has been activated, the copy can be accessed via the VDEV target device immediately.) Unless the data is changed by the host accessing the VDEV, the VDEV always presents a frozen point-in-time copy of the source device at the point of activation. Data can also be restored from the virtual devices back to the source devices. When the information is no longer needed, the session can be terminated.

TimeFinder/Snap operations are controlled from the host by using the `symsnap` command to create, activate, restore, and terminate TimeFinder/Snap copy sessions.

Figure 48 on page 192 illustrates a virtual copy session where the controlling host creates a copy of standard device DEV001 on target device VDEV005.
The `sym snap` command is used to enable TimeFinder/Snap operations. The snap operation happens in two phases: creation and activation. The creation phase builds bitmaps of the source and target that are later used to manage the changes on the source and target. The creation of a snap pairing does not copy the data from the source volume to the target volume. To create snap sessions on all the standards and BCVs in the device group `MyDevGrp`, use the following command.

```
sym snap -g <MyDevGrp> create -noprompt
```

The activation of a snap session enables the protection of the source data tracks. When protected tracks are changed on the source volume, they are first copied into the save pool and the VDEV pointers are updated to point to the changed tracks in the save pool. When tracks are changed on the VDEV, the data is written directly to the save pool and the VDEV pointers are updated in the same way.

Activation of the snap session created in the previous `create` command can be accomplished by executing the following command.

```
sym snap -g MyDevGrp activate -noprompt
```
Performing TimeFinder/Mirror operations

Symmetrix TimeFinder/Mirror is essentially a business continuance solution that allows the use of special business continuance volume (BCV) Symmetrix devices. Copies of data from a standard Symmetrix device (which are online for regular I/O operations from the host) are sent and stored on BCV devices to mirror the primary data. Uses for the BCV copies can include backup, restore, decision support, and applications testing. Each BCV device has its own host address, and is configured as a stand-alone Symmetrix device.

A business continuance sequence first involves associating and establishing the BCV device as a mirror of a specific standard Symmetrix device. As a result, the BCV device becomes inaccessible (Not Ready) using its original device address while it is in an established pair. Once the BCV device is synchronized, you can separate (split) it from the standard device with which it is paired, thereby making it available again to its host for backup or other host processes through its original device address.

After host processing on the BCV device is complete, the BCV may again be mirrored to a standard Symmetrix device — either the same device with which it was previously paired, or with a different device.

Note: For Symmetrix configurations running Enginuity release level 5874 and Solutions Enabler 7.0, the TimeFinder/Mirror functions described herein will be performed through TimeFinder/Clone software using a process called Clone Emulation. Clone Emulation mode makes the use of RAID-protected BCVs transparent to the TimeFinder/Mirror user.

For backward compatibility, TimeFinder/Clone Emulation mode transparently performs TimeFinder/Mirror commands and executes scripts written for Solutions Enabler up through version 6.5.2 running on Symmetrix arrays using Enginuity release levels 5773 and earlier.

TimeFinder/Mirror Establish operations

A BCV device can be fully or incrementally established. After configuration and initialization of a Symmetrix system, BCV devices contain no data. Like standard devices, BCV devices can have unique host addresses and can be online and ready to the host(s) to which they are connected. A full establish operation must be used the
first time the standard devices are paired with BCV devices. An incremental establish of a BCV device can be performed to resynchronize any data that has changed on the standard since the last establish operation.

**Note:** When BCVs are established, they are not accessible to a host.

Symmetrix systems allow up to four mirrors for each logical volume. The mirror positions are commonly designated M1, M2, M3, and M4. An unprotected BCV can be the second, third, or fourth mirror position of the standard device. A host, however, logically views the Symmetrix M1/M2 mirrored devices as a single device.

In order to perform an establish operation on a BCV device using the SYMCLI, the BCV device must be associated with an existing device or composite group (this is not a requirement when using a device file). You can associate BCV devices with a device/composite group by using either the physical device name, or the Symmetrix device name. For example, to associate a BCV device with a physical device name of `/dev/dsk/c0t2d0s2` to a device group named `prod` (and assign the name `BCV7` to the BCV device), you would execute a command that looks like this:

```
symbcv -g prod associate pd c0t2d0 BCV7
```

On the other hand, to associate a BCV device with a Symmetrix device name of `001F` to a device group named `prod` (and assign the name `BCV7` to the BCV device), you would execute a command that looks like this instead:

```
symbcv -g prod associate dev 001F BCV7
```

To assign a BCV as a mirror of a standard Symmetrix device, the BCV pair must be established. This is done by executing the `symmir establish` command. One method of establishing a BCV pair is to allow the standard/BCV device-pairing algorithm to arbitrarily create BCV pairs from multiple devices within a device group. For example:

```
symmir -g MyDevGrp establish -full -noprompt
```

With this method, TimeFinder/Mirror first checks for any attach assignments (specifying a preferred BCV match from among multiple BCVs in a device group). TimeFinder/Mirror then checks if there are
any pairing relationships among the devices. If either of these previous conditions exists, TimeFinder/Mirror uses these assignments.

Incrementally establishing a BCV pair accomplishes the same thing as a full establish process, with a major time-saving exception: the standard device copies to the BCV device only the new data that was updated on the standard device while the BCV pair was split. Any changed tracks on the BCV are also overwritten by the data on the corresponding tracks on the standard device.

**TimeFinder/Mirror split operations**

Splitting a BCV pair is a TimeFinder/Mirror action that detaches the BCV from its standard device and makes the BCV ready for host access. When splitting a BCV, the system must perform housekeeping tasks that may require a few milliseconds on a busy Symmetrix system. These tasks involve a series of steps that result in separation of the BCV from its paired standard:

- I/O is suspended briefly to the standard device.
- Write pending tracks for the standard device that have not yet been written out to the BCV are duplicated in cache to be written to the BCV.
- The BCV is split from the standard device.
- The BCV device status is changed to “Ready”.

**Regular split**

A regular split is the type of split that has existed for TimeFinder/Mirror since its inception. With a regular split (before Enginuity version 5568), I/O activity from the production hosts to a standard volume was not accepted until it was split from its BCV pair. Therefore, applications attempting to access the standard or the BCV would experience a short wait during a regular split. Once the split was complete, no further overhead was incurred.

**Instant split**

An instant split shortens the wait period during a split by dividing the process into a foreground split and a background split. During an instant split, the system executes the foreground split almost instantaneously and returns a successful status to the host. This instantaneous execution allows minimal I/O disruptions to
the production volumes. Furthermore, the BCVs are accessible to the
hosts as soon as the foreground process is complete. The background
split continues to split the BCV pair until it is complete. When the
-instant option is included or defaulted, SYMCLI returns
immediately after the foreground split, allowing other operations
while the BCV pair is splitting in the background.

Beginning with Enginuity version 5568, any split operation is an
instant split. A regular split is still valid for earlier versions and
for current applications that perform regular split operations.
However, current applications that perform regular splits with
Enginuity version 5568 actually perform an instant split.

By specifying the -instant option on the command line, an instant
split with Enginuity versions 5x66 and 5x67 can be performed.
Since version 5568, this option is no longer required because instant
split mode has become the default behavior. It is beneficial to
continue to supply the -instant option with later Enginuity
versions, otherwise the default is to wait for the background split
to complete.

For example, to perform an instant split on all BCV pairs in a device
group named MyDevGrp, and allow SYMCLI to return to the server
process while the split is performed in the background, you would
execute a command that looks like this:

```
symmir -g MyDevGrp split -instant -noprompt
```

And once the split operation is started, the progress of the split can be
obtained by executing a symmir query command that looks like
this:

```
symmir -cg MyConGrp query -bg
```

In this example, the -bg option is provided to query the status of the
background split.

**TimeFinder/Mirror restore operations**

Once established, a BCV device can be used to fully or incrementally
restore data on its associated standard devices. Like the full
establish operation, a full restore operation copies the entire
contents of the BCV devices to the standard devices. The devices
upon which the restore operates may be defined in a device group,
composite group, or device file. For example:
symmir -g MyDevGrp -full restore -noprompt
symmir -cg MyConGrp -full restore -noprompt
symmir -f MyFile -full -sid 109 restore -noprompt

The incremental restore process accomplishes the same thing as the full restore process with a major time-saving exception. The BCV copies to the standard device only new data that was updated on the BCV device while the BCV pair was split. The data on the corresponding tracks of the BCV device also overwrites any changed tracks on the standard device. This maximizes the efficiency of the resynchronization process. This process is useful, for example, if, after testing or validating an updated version of a database or a new application on the BCV device is completed, a user wants to migrate and utilize a copy of the tested data or application on the production standard device.

Note: An incremental restore of a BCV volume to a standard volume is only possible when the two volumes have an existing TimeFinder relationship.

TimeFinder consistent split

TimeFinder consistent split allows you to split off a dependent-write consistent, restartable image of an application without interrupting online services. Consistent split helps to avoid inconsistencies and restart problems that can occur when splitting an application-related BCV without first quiescing or halting the application. Consistent split is implemented using Enginuity Consistency Assist (ECA) feature.

Enginuity Consistency Assist

The Enginuity Consistency Assist (ECA) feature of the Symmetrix operating environment can be used to perform consistent split operations across multiple heterogeneous environments. This functionality requires a TimeFinder/CG license and uses the -consistent option of the symmir command.

Using ECA to consistently split BCV devices from the standards, a control host with no database or a database host with a dedicated channel to gatekeeper devices must be available. The dedicated channel cannot be used for servicing other devices or to freeze I/O. For example, to split a device group, execute:
symmir -g MyDevGrp split -consistent -noprompt

Figure 49 illustrates an ECA split across three database hosts that access devices on a Symmetrix system.

Device groups or composite groups must be created on the controlling host for the target application to be consistently split. Device groups can be created to include all of the required devices for maintaining business continuity. For example, if a device group is defined that includes all of the devices being accessed by Hosts A, B, and C (see Figure 49), then all of the BCV pairs related to those hosts can be consistently split with a single command.

However, if a device group is defined that includes only the devices accessed by Host A, then the BCV pairs related to Host A can be split without affecting the other hosts. The solid vertical line in Figure 49 represents the ECA holding of I/Os during an instant split process, creating a dependent-write consistent image in the BCVs.

Figure 50 on page 199 illustrates the use of local consistent split with a database management system (DBMS).
When a **split** command is issued with ECA from the production host, a consistent database image is created using the following sequence of events shown in Figure 50:

1. The device group, device file, or composite group identifies the standard devices that hold the database.
2. SYMAPI communicates to Symmetrix Enginuity to validate that all identified BCV pairs can be split.
3. SYMAPI communicates to Symmetrix Enginuity to open the ECA window (the time within Symmetrix Enginuity where the writes are deferred), the instant split is issued, and the writes are released by closing the window.
4. ECA suspends writes to the standard devices that hold the database. The DBMS cannot write to the devices and subsequently waits for these devices to become available before resuming any further write activity. Read activity to the device is not affected unless attempting to read from a device with a write queued against it.
5. SYMAPI sends an instant split request to all BCV pairs in the specified device group and waits for the Symmetrix to acknowledge that the foreground split has occurred. SYMAPI then communicates with Symmetrix Enginuity to resume the write or close the ECA window.
6. The application resumes writing to the production devices.

The BCV devices now contain a restartable copy of the production data that is consistent up until the time of the instant split. The production application is unaware that the **split** or
suspend/resume operation occurred. When the application on the secondary host is started using the BCVs, there is no record of a successful shutdown. Therefore, the secondary application instance views the BCV copy as a crashed instance and proceeds to perform the normal crash recovery sequence to restart.

When performing a consistent split, it is a good practice to issue host-based commands that commit any data that has not been written to disk before the split to reduce the amount of time on restart. For example on UNIX systems, the sync command can be run; from a database perspective, a checkpoint or equivalent should be executed. (For DB2, this involves temporarily suspending, and subsequently resuming database write operations.)

----

TimeFinder/Mirror reverse split

BCVs can be mirrored to guard against data loss through physical drive failures. A reverse split is applicable for a BCV that is configured to have two local mirrors. It is generally used to recover from an unsuccessful restore operation. When data is restored from the BCV to the standard device, any writes that occur while the standard is being restored alter the original copy of data on the BCVs primary mirror. If the original copy of BCV data is needed again at a later time, it can be restored to the BCVs primary mirror from the BCVs secondary mirror using a reverse split. For example, whenever logical corruption is reintroduced to a database during a recovery process (following a BCV restore), both the standard device and the primary BCV mirror are left with corrupted data. In this case, a reverse split can restore the original BCV data from a BCVs secondary mirror to its primary mirror.

This is particularly useful when performing a restore and immediately restarting processing on the standard devices when the process may have to be restarted many times.

**Note:** Reverse split is not available when protected restore is used to return the data from the BCVs to the standards.
A traditional DB2 database back up operation can require a significant amount of processing overhead, both at the server and on the storage system used. And, depending upon the size of the database being backed up and the type of back up image being created (full, incremental, or delta), a single back up operation can take from just a few minutes to several hours to complete. (Starting with DB2 9, the performance impact of a back up operation can be controlled somewhat through the use of utility throttling; however, throttled back up operations usually run longer than unthrottled back up operations.) Because of this, in most production environments, back up operations are scheduled during off-peak hours and frequent back up operations are not always possible.

By using EMC TimeFinder technology, a database administrator can develop a database back up (and recovery) strategy that meets the ever-increasing demands for availability, while minimizing the impact on servers, storage systems, and operations staff. This chapter is designed to provide you with guidelines for using TimeFinder technology to back up DB2 LUW databases that reside on EMC Symmetrix storage arrays.

Topics covered include:

- Using disk-based replication to back up a DB2 LUW database
- Planning for disk-based replication recovery
- Backing up a DB2 LUW database that has been taken offline
- Backing up a DB2 LUW database while it remains online
- Backing up the database copy
Note: The procedures outlined in this chapter make the assumption that all necessary Symmetrix standard devices (STDs), Business Continuance Volume devices (BCVs), clone target devices (TGTs), virtual devices (VDEVs), and save devices (SAVDEVs) have been created and added to a device group.
Using disk-based replication to back up a DB2 LUW database

EMC disk-based replication can be used to copy a DB2 LUW database and the copy produced, in turn, can act as a database back up image that can be used to restore the database in the event it becomes damaged or corrupted. However, when disk-based replication is used to back up a database, DB2’s recovery history file is not updated to reflect that a back up image has been created. Therefore, when disk-based replication is used for back up purposes, it must also be used for recovery; traditional DB2 back up and recovery methods no longer apply. (However, traditional back up methods can be used to create a back up image from the database copy once the copy is made.)

One of the easiest ways to create a disk-based replication copy of a DB2 LUW database stored on a Symmetrix storage array is to use the EMC local replication product, TimeFinder. The TimeFinder family of products consists of TimeFinder/Clone and TimeFinder/Snap; all three products can be used to create a database copy that can be used as a back up image. The actual process used is dependent upon the type of TimeFinder technology used and the state the database is in at the time the copy is made. A back up copy of a database can be made at any time; however, if the database is online, it must be placed in “write-suspend” mode before replication takes place. This ensures that all database-related files are flushed to disk before their blocks are duplicated. (Once a back up copy has been made, the database must be taken out of “write-suspend” mode before disk I/O can continue.)

IMPORTANT

When disk-based replication is used to back up a DB2 database, only full database backup images can be created; the creation of incremental and delta backup images is not supported. However, because the use of disk-based replication improves the performance of backup operations, incremental and delta database backup images do not offer any real advantages.
### Planning for disk-based replication recovery

If you plan on using disk-based replication for back up and recovery, this must be taken into consideration when deciding on the physical layout to use for the database. Since disk-based replication copies volumes at the physical disk level (as seen by the host), all table space containers for a database should reside on a set of disks that are dedicated to the database and are not shared with other applications and databases. (For Linux and UNIX systems, ensure that the containers used reside in a volume group that is dedicated to the database.) Sharing disks with other applications can cause unnecessary work for the array and wasted space on the copy target volumes.

In addition to isolating a database to its own set of dedicated volumes, storage for the database should also be divided into two parts: one for the database and its associated data objects and the other for the database's transaction logs. Such a division allows the two parts to be manipulated independently of each other in the event a recovery operation needs to be performed. In fact, this division is crucial if the database is to be a recoverable database. (A database is considered nonrecoverable when both its `logarchmeth1` and `logarchmeth2` configuration parameters are set to **OFF**, in which case, circular logging is used and roll-forward recovery is not possible. If either of these parameters is assigned a value other than **OFF**, archival logging is used and the database is considered to be recoverable since crash recovery, version recovery, and roll-forward recovery are possible. For more information, refer to “Recoverable and nonrecoverable databases” on page 144.)

Why is such a division important for a recoverable database? Because if both the data objects and the recovery objects reside on the same physical volume, both the transaction log files and the database’s data files will be returned to the state they were in at the time the back up copy was made if a disk-based replication restore operation is performed. As a result, roll-forward recovery will no longer be possible since every transaction log record generated after the database copy was made will be erased by the restore process. By separating data objects from recovery objects, a database can be restored without impacting its transaction log files. Once the database has been restored, records stored in its transaction log files can be reapplied to return the database to the state it was in at a specific point in time.
To facilitate database backup and recovery with TimeFinder technology, it is also a good idea to create two different SYMCLI device or composite groups. (Device groups are used if the database resides on a single Symmetrix frame; composite groups are used if the database spans more than one Symmetrix frame.) Ideally, one device or composite group should encompass all volumes used by the database and the other device or composite group should include just the volumes that contain the database and its associated data objects. (In order for a device to be included in more than one device or composite group, the `SYMAPI_ALLOW_DEV_IN_MULT_GRPS` option in the SYMCLI options file must be set to `ENABLE`.) When this approach is used, backup operations can be performed using the device or composite group that encompasses the entire database while recovery operations can be performed using either the same the device or composite group (if the database is a nonrecoverable database) or the device or composite group that includes just the volumes that contain the database and its associated data objects (if the database is a recoverable database).

Figure 51 on page 206 depicts a configuration where a DB2 database’s data objects and recovery objects have been separated into their own physical volumes and those volumes, in turn, have been organized into two separate SYMCLI device or composite groups.
Figure 51  Configuration to facilitate TimeFinder backup and recovery

Note: In the case of multipartitioned databases, all devices from all database partitions should be included in the device/composite groups created.
Backing up a DB2 LUW database that has been taken offline

Ideally, a copy of a DB2 LUW database should be taken while the database is offline. (A DB2 LUW database is considered offline when all connections to it have been terminated and the database itself is no longer active; if a DB2 LUW database was brought online by the execution of the `ACTIVATE DATABASE` command, it must be taken offline by executing the `DEACTIVATE DATABASE` command.) In most cases, this ensures that the database is in a consistent state at the time the copy is made. This is important because in order to restore a database that has become damaged or corrupted, you must have a clean, consistent copy or back up image of the database available.

The following sections describe how to create a disk-based replication back up image of a DB2 LUW database using EMC TimeFinder technology, while the database is offline. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.

Creating an offline database back up image with TimeFinder/Clone

TimeFinder/Clone can be used to create a point-in-time copy of a database, which can then be used for back up operations, decision support, data warehouse refreshes, or any other process that requires parallel access to production information. With TimeFinder/Clone, up to 16 simultaneous and immediately available full-size target copies (sessions) of source devices can be created. Both source and target devices can be any combination of standard or BCV devices and can have different RAID protection; however, the sizes of the source and target devices used have to match. The relationship between a clone source and target is defined when a clone session is created. Once the session is later activated, the target device can be instantly accessed by a target’s host — when a session is activated, copying to the clone can take place immediately or it can be deferred (CopyOnAccess). Clone target devices can be later refreshed from the source devices (re-activated) or can rewrite the source devices with their content (restored), or the clone session between source and target devices can be terminated.
A full copy will take place only during the first time a clone session is created. Any further operations on the clone, such as re-create or restore, will be done incrementally, passing just the final changes between the source and target devices, making TimeFinder/Clone a very fast and scalable database cloning solution, regardless of the database size. In addition, as soon as TimeFinder activates or restores a clone session, control is returned almost immediately to the user, although data copy may be still proceeding in the background. To a host accessing the devices, the data on them will seem as if the operation has already completed. These abilities make TimeFinder/Clone incredibly fast in creating or restoring database replicas of any size.

Note: Before a target device can be cloned from a source device, both the source device and the target device must have been previously associated with a device or composite group. Additionally, the target and source devices must be of the same size and emulation type.

To create an offline back up copy of a DB2 database using TimeFinder/Clone, complete the following steps, in the order shown:

1. Create a relationship between the Symmetrix devices that contain the database to be backed up and the Symmetrix devices that are to serve as the target for the clone by executing the following command at the host (as the system administrator or root user):

   ```
   symclone -g [DeviceGroup] create -differential -tgt -noprompt
   ```

   or

   ```
   symclone -g [DeviceGroup] create -precopy -differential -tgt -noprompt
   ```

   or

   ```
   symclone -g [DeviceGroup] create -copy -differential -tgt -noprompt
   ```

   where:

   • *DeviceGroup* — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to create a clone of a set of standard Symmetrix devices that have been assigned to a device group named *symdb2dg*, you would execute a command that looks like this:
Backing Up a DB2 LUW Database Using TimeFinder Technology

**symclone** -g symdb2dg create -copy -differential -tgt -noprompt

And when this command is executed, you should see a message that looks something like this:

'Create' operation execution is in progress for device group 'symdb2dg'. Please wait...

'Create' operation successfully executed for device group 'symdb2dg'.

In this example, data will be copied to the target devices in the background when the clone is activated because the **-copy** keyword was specified with the **symclone create** command used. If you want data to be copied to the target devices in the background when the clone is created, specify the **-precopy** keyword instead of the **-copy** keyword; when the clone is activated the **-precopy** behavior will change to **-copy** behavior and any blocks that have changed since the clone was created will be re-copied to the target devices. If you do not want data to be copied when the clone is created and/or activated, do not specify the **-precopy** or **-copy** keyword with the **symclone create** command used.

2. Verify that the clone was created by executing the following command at the host (as the system administrator or root user):

**symclone list**

When this command is executed, you should see a message that looks something like this:

<table>
<thead>
<tr>
<th>Symmetrix ID: 000190300709</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Device</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
<tr>
<td>Sym</td>
</tr>
<tr>
<td>----</td>
</tr>
<tr>
<td>0600</td>
</tr>
<tr>
<td>060A</td>
</tr>
<tr>
<td>0614</td>
</tr>
<tr>
<td>061E</td>
</tr>
<tr>
<td>Total Tracks</td>
</tr>
<tr>
<td>MB(s)</td>
</tr>
</tbody>
</table>

Legend:
3. Terminate all connections to the database by executing one of the following commands at the host (as the database instance user):

   **DEACTIVATE DATABASE [DBAlias]**

   where:

   - **DBAlias** — Identifies the alias of the database to be deactivated (stopped).

   or

   **FORCE APPLICATION ALL**

   or

   **db2stop FORCE**

   For multipartition databases, the **DEACTIVATE DATABASE** command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

   **CAUTION**

   The **FORCE APPLICATION ALL** command will terminate all connections to all databases under the current DB2 Database Manager instance’s control; the **db2stop FORCE** command will terminate all connections to all databases under the current instance’s control and then stop the instance. (If the **db2stop FORCE** command is issued and applications do not respond within the time frame specified by the **start_stop_time** DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance’s control.
4. Activate the clone by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] activate -consistent -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

Note: The **-consistent** keyword tells Enginuity to use Enginuity Consistency Assist (ECA) to momentarily suspend writes to the source disks while the clone is being activated.

For example, to activate a clone that was created earlier for a device group named `symdb2dg`, you would execute a command that looks like this:

```
symclone -g symdb2dg activate -consistent -tgt -noprompt
```

And when this command is executed, you should see a message that looks something like this:

'Activate' operation execution is in progress for device group 'symdb2dg'. Please wait...

'Activate' operation successfully executed for device group 'symdb2dg'.

5. Verify that data in the source devices used is being copied to the appropriate target devices by executing one of the following commands at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] verify -copied
```

or

```
symclone -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to determine whether data is being copied to the set of clone target devices that have been assigned to a device group named `symdb2dg`, you could execute a command that looks like this:
### Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

#### Backing Up a DB2 LUW Database Using TimeFinder Technology

The command to back up a DB2 LUW database using TimeFinder technology is:

```
symclone -g symdb2dg query
```

When this command is executed, you should see a message that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type             : REGULAR
DG's Symmertix ID     : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV001</td>
<td>TGT001</td>
<td>0</td>
<td>XXX.</td>
</tr>
<tr>
<td>DEV002</td>
<td>TGT002</td>
<td>0</td>
<td>XXX.</td>
</tr>
<tr>
<td>DEV003</td>
<td>TGT003</td>
<td>0</td>
<td>XXX.</td>
</tr>
<tr>
<td>DEV004</td>
<td>TGT004</td>
<td>0</td>
<td>XXX.</td>
</tr>
</tbody>
</table>

Total: Track(s) 2216731, MB(s) 138546

Legend:
- **(C):** X = The background copy setting is active for this pair.
  . = The background copy setting is not active for this pair.
- **(G):** X = The Target device is associated with a group.
  . = The Target device is not associated with a group.
- **(D):** X = The Clone session is a differential copy session.
  . = The Clone session is not a differential copy session.
- **(P):** X = The pre-copy operation has completed one cycle
  . = The pre-copy operation has not completed one cycle
```

6. Once the clone has been successfully activated, you can re-establish connections to the database stored on the source Symmetrix devices by executing one of the following commands at the host (as the database instance user):

   **ACTIVATE DATABASE** [DBAlias]

   where:

   - **DBAlias** — Identifies the alias of the database to be activated.

   or

   **CONNECT TO** [DBServer]

   where:

   - **DBServer** — Identifies the database server or the alias of the database to which an application or user is to connect.
Note: If the command `db2stop FORCE` was used to terminate all connections to the database, the command `db2start` must be executed before the `ACTIVATE DATABASE` or `CONNECT` command will execute successfully.

After completing these steps, you will have a point-in-time copy of the database (on the clone target devices) that can be used to restore the database if necessary.

It is important to note that databases that have been copied using TimeFinder/Clone are subject to Copy On First Write (COFW) and Copy On Access (COA) penalties when Enginuity 5771 and earlier is used. The COFW penalty arises because the first time a track is written to the source device, if it has not yet been copied to the target device, it must first be copied to the target device before the write from the host will be acknowledged. Subsequent writes to tracks that have already been copied do not suffer this penalty. With Enginuity 5772 and later, COFW is performed asynchronous to the host and has minimal impact on write performance.

The COA penalty arises if a track on a target volume is accessed before it has been copied, requiring it to first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems. The COA affect can be mitigated by specifying the `-precopy` or `-copy` keyword with the `symclone create` command used.

Creating an offline database back up image with TimeFinder/Snap

TimeFinder/Snap software allows users to make multiple pointer-based, space-saving copies of data simultaneously on multiple target devices using a single source device. The resulting point-in-time copy of data is available to the target host for access instantly. TimeFinder/Snap allows data to be copied from a single source device to as many as 128 target devices simultaneously, where the source devices can be either a Symmetrix standard device or a business continuance volume (BCV) device, with the exception being a BCV working in TimeFinder/Clone's clone emulation mode. The target devices are Symmetrix virtual devices (VDEVs), which consume only a fraction of the disk space required by the original information because they use pointers to track changed data. (In
order to support 128 different target devices, the
SYMCLI_MULTI_VIRTUAL_SNAP environment variable must be set to ENABLE.)

A VDEV is a host-addressable Symmetrix device with special attributes. Unlike a BCV or clone copy, which contains a full volume of data, the VDEV is a logical-image device that offers a space-saving way to create instant point-in-time copies of logical volumes. These copies are not full copies of the data; instead, they are logical images of the original information, as it existed at the time the virtual copy was created. Any updates to a source device after its activation with a virtual device cause the preupdate image of the changed tracks to be copied to specifically created target volumes within the Symmetrix referred as save devices (SAVDEVs). These save devices are aggregated into save device pools, to which VDEVs are assigned. The virtual device’s indirect pointer is then updated to point to the original track data on the save device, preserving a point-in-time image of the volume. TimeFinder/Snap uses this copy-on-first-write (COFW) technique to conserve disk space; only changes to tracks on the source cause incremental storage to be consumed.

TimeFinder/Snap functionality is controlled via copy sessions, which pair source and target devices. Sessions are maintained on the Symmetrix array and can be queried to verify the current state of the devices. A copy session must first be created, and then activated, for the target virtual devices to become accessible to their host. Unless the data is changed by the host accessing the virtual device, the virtual device always presents a frozen point-in-time copy of the source device at the point of activation. When the copy session information is no longer required, the copy session can be terminated.

In those scenarios where multiple replicas of a source environment are required, TimeFinder/Snap provides additional benefits in storage allocation reduction by utilizing a common save pool strategy. As multiple sessions are activated simultaneously or at different times, by virtue of using pointers only a single copy of shared data is required, therefore all TimeFinder/Snap sessions sharing changed data will point to this same common block. Thus, whether a single TimeFinder/Snap session has been activated against a single source, or 128, all track pointers sharing data in the pool will point to the single copy of that data in the save pool, greatly reducing the amount of additional capacity required to support multiple point-in-time images.
To create an offline back up copy of a DB2 database using TimeFinder/Snap, complete the following steps, in the order shown:

1. Create a relationship between the Symmetrix devices that contain the database to be backed up and the VDEVs and SAVDEVs that are to serve as the target for the virtual device copy and any related data changes by executing the following command at the host (as the system administrator or root user):

   ```sh
   symsnap -g [DeviceGroup] create -svp [SavePool] -noprompt
   ```

   where:

   - **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
   - **SavePool** — Identifies the SAVDEVs that are to be used to store original tracks associated with the virtual device copy that are changed once the copy session is activated.

   For example, to create a copy session relationship for a set of Symmetrix devices that have been assigned to a device group named `symdb2dg`, execute a command that looks like this:

   ```sh
   symsnap -g symdb2dg create -svp DB2_snappool -noprompt
   ```

   And when this command is executed, you should see a message that looks something like this:

   'Create' operation execution is in progress for device group 'symdb2dg'. Please wait...

   'Create' operation successfully executed for device group 'symdb2dg'.

2. Verify that the copy session relationship was created by executing the following command at the host (as the system administrator or root user):

   ```sh
   symsnap list
   ```

   When this command is executed, you should see a message that looks something like the following:
### Backing Up a DB2 LUW Database Using TimeFinder Technology

Symmetrix ID: 000190300709

<table>
<thead>
<tr>
<th>Sym</th>
<th>Sym</th>
<th>Tracks</th>
<th>Sym</th>
<th>G SRC &lt;=&gt; TGT</th>
<th>PoolName</th>
</tr>
</thead>
<tbody>
<tr>
<td>0600</td>
<td>05C0</td>
<td>655500</td>
<td>X</td>
<td>Created</td>
<td>DB2_snappool</td>
</tr>
<tr>
<td>060A</td>
<td>05CA</td>
<td>655500</td>
<td>X</td>
<td>Created</td>
<td>DB2_snappool</td>
</tr>
<tr>
<td>0614</td>
<td>05D4</td>
<td>655500</td>
<td>X</td>
<td>Created</td>
<td>DB2_snappool</td>
</tr>
<tr>
<td>061E</td>
<td>05DE</td>
<td>655500</td>
<td>X</td>
<td>Created</td>
<td>DB2_snappool</td>
</tr>
</tbody>
</table>

Total Tracks: 2949669

MB(s): 184354

**Legend:**

(G): X = The Target device is associated with a group,

. = The Target device is not associated with a group.

3. Terminate all connections to the database by executing one of the following commands at the host (as the database instance user):

```
DEACTIVATE DATABASE [DBAlias]
```

where:

- `DBAlias` — Identifies the alias of the database to be deactivated (stopped).

or

```
FORCE APPLICATION ALL
```

or

```
db2stop FORCE
```

For multipartition databases, the `DEACTIVATE DATABASE` command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

**CAUTION**

The `FORCE APPLICATION ALL` command will terminate all connections to all databases under the current DB2 Database Manager instance’s control; the `db2stop FORCE` command will terminate all connections to all
databases under the current instance’s control and then stop the instance. (If the `db2stop FORCE` command is issued and applications do not respond within the time frame specified by the `start_stop_time` DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance’s control.

4. Activate the copy session relationship by executing the following command at the host (as the system administrator or root user):

   ```
   symsnap -g [DeviceGroup] activate -consistent -noprompt
   ```

   where:

   - `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

   Note: The `-consistent` keyword tells Enginuity to use Enginuity Consistency Assist (ECA) to momentarily suspend writes to the source disks while the copy session relationship is being activated.

   For example, to activate a copy session relationship that was created earlier for a device group named `symdb2dg`, you would execute a command that looks like this:

   ```
   symsnap -g symdb2dg activate -consistent -noprompt
   ```

   And when this command is executed, you should see a message that looks something like this:

   'Activate' operation execution is in progress for device group 'symdb2dg'. Please wait...

   'Activate' operation successfully executed for device group 'symdb2dg'.

5. Verify that the copy session relationship has been activated by executing the following command at the host (as the system administrator or root user):

   ```
   symsnap -g [DeviceGroup] query
   ```

   where:
DeviceGroup — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine whether a copy session relationship for the set of Symmetrix devices that have been assigned to a device group named symdb2dg has been successfully activated, you would execute a command that looks like this:

```
symsnap -g symdb2dg query
```

And if the copy session relationship was successfully activated, you should see a message that looks something like this:

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td>Changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>Sym</td>
<td>Tracks</td>
<td>Sym</td>
<td>G</td>
</tr>
<tr>
<td>G</td>
<td>Tracks</td>
<td>SRC &lt;&gt; TGT (%)</td>
<td></td>
</tr>
</tbody>
</table>

| DEV001 | 0600 | 655499 | VDEV001 | 05C0 | X | 0 | CopyOnWrite | 0 |
| DEV002 | 060A | 655499 | VDEV002 | 05CA | X | 0 | CopyOnWrite | 0 |
| DEV003 | 0614 | 655499 | VDEV003 | 05D4 | X | 0 | CopyOnWrite | 0 |
| DEV004 | 061E | 655499 | VDEV001 | 05DE | X | 0 | CopyOnWrite | 0 |

**Total**

Track(s):

1966497

MB(s):

122906

Legend:

(G): X = The Target device is associated with this group,
     . = The Target device is not associated with this group.

6. Once the activate operation is complete, you can re-establish connections to the database by executing one of the following commands at the host (as the database instance user):

```
ACTIVATE DATABASE [DBAlias]
```

where:

- **DBAlias** — Identifies the alias of the database to be activated.

or

```
CONNECT TO [DBServer]
```

where:
• **DBServer** — Identifies the database server or the alias of the database to which an application or user is to connect.

**Note:** If the command `db2stop FORCE` was used to terminate all connections to the database, the command `db2start` must be executed before the `ACTIVATE DATABASE` or `CONNECT` command will execute successfully.

After completing these steps, you will have a point-in-time copy of the database (in the form of a virtual device copy) that can be used to restore the database if necessary.

**Note:** Databases that have been copied using TimeFinder/Snap are subject to Copy On First Write (COFW) penalties when the copy session is activated if Enginuity 5771 and earlier are used. The COFW penalty arises because the first time a track is written to the source device if it has not yet been copied to the appropriate SAVDEV, it must first be copied to the SAVDEV before the write from the host will be acknowledged. With Enginuity 5772 and later, COFW is performed asynchronously to the host and has minimal impact on write performance.
Backing up a DB2 LUW database while it remains online

In environments where it is not always possible to take a DB2 LUW database offline in order to back it up, an alternative approach is to make a copy of the database while it remains online. As noted in Chapter 3, “DB2 for Linux, UNIX, and Windows Back Up and Recovery Concepts,” DB2 provides a way to temporarily suspend (and later resume) all database I/O so that a database can be backed up (via split mirroring) without having to be taken offline. The command that provides this functionality is the `SET WRITE` command, which was introduced in DB2 for LUW 7.1 Fix Pack 2.

When executed, the command `SET WRITE SUSPEND FOR DATABASE` causes DB2 to suspend all write operations to table space containers and log files that are associated with the current database. (The suspension of writes to table spaces and log files is intended to prevent partial page writes from occurring until the suspension is removed.) All operations, apart from online back up and restore operations, will function normally while database writes are suspended. That's because read-only transactions are not suspended and are able to continue working with a database that has been placed in a “Write-Suspended” state provided they do not require write processing; applications can continue to perform insert, update, and delete operations with data that has been cached in the database’s buffer pool(s), but new pages cannot be written to disk. Additionally, new database connections can be established to a write-suspended database provided there is enough room in a buffer pool to hold the system catalog pages needed to authenticate the connections desired.

Write I/O operations for a write-suspended database can be resumed at any time by executing the command `SET WRITE RESUME FOR DATABASE`. When executed, this command causes DB2 to lift all write suspensions and allow write operations to table space containers and log files that are associated with the current database to continue.

The following sections describe how to create a disk-based replication back up image of a DB2 LUW database, using EMC TimeFinder technology, while the database is online by momentarily placing it in a “Write-Suspended” state. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.
Creating an online database back up image with TimeFinder/Clone

To create an online back up copy of a DB2 database using TimeFinder/Clone, complete the following steps, in the order shown:

1. Create a relationship between the Symmetrix devices that contain the database to be backed up and the Symmetrix devices that are to serve as the target for the clone by executing the following command at the host (as the system administrator or root user):

   ```
   symclone -g [DeviceGroup] create -differential -tgt -noprompt
   or
   symclone -g [DeviceGroup] create -precopy -differential -tgt -noprompt
   or
   symclone -g [DeviceGroup] create -copy -differential -tgt -noprompt
   ```

   where:
   - `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

   For example, to create a clone of a set of standard Symmetrix devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

   ```
   symclone -g symdb2dg create -copy -differential -tgt -noprompt
   ```

   And when this command is executed, you should see a message that looks something like this:

   'Create' operation execution is in progress for device group 'symdb2dg'. Please wait...
   
   'Create' operation successfully executed for device group 'symdb2dg'.

   In this example, data will be copied to the target devices in the background when the clone is activated because the `-copy` keyword was specified with the `symclone create` command used. If you want data to be copied to the target devices in the background when the clone is created, specify the `-precopy` keyword instead of the `-copy` keyword; when the clone is activated the `-precopy` behavior will change to `-copy` behavior.
and any blocks that have changed since the clone was created will be re-copied to the target devices. If you do not want data to be copied when the clone is created and/or activated, do not specify the `--precopy` or `--copy` keyword with the `symclone create` command used.

2. Verify that the clone was created by executing the following command at the host (as the system administrator or root user):

```bash
symclone list
```

When this command is executed, you should see a message that looks something like this:

```
Symmetrix ID: 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sym</td>
<td>Sym</td>
<td></td>
</tr>
<tr>
<td>Tracks</td>
<td>Sym</td>
<td>CGDP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SRC &lt;=</td>
</tr>
<tr>
<td>0600</td>
<td>655500</td>
<td>070F</td>
</tr>
<tr>
<td>060A</td>
<td>655500</td>
<td>0719</td>
</tr>
<tr>
<td>0614</td>
<td>655500</td>
<td>0723</td>
</tr>
<tr>
<td>061E</td>
<td>655500</td>
<td>072D</td>
</tr>
</tbody>
</table>

Legend:

(C): X = The background copy setting is active for this pair.
. = The background copy setting is not active for this pair.
(G): X = The Target device is associated with a group.
. = The Target device is not associated with a group.
(D): X = The Clone session is a differential copy session.
. = The Clone session is not a differential copy session.
(P): X = The pre-copy operation has completed one cycle
. = The pre-copy operation has not completed one cycle
```

3. Establish a connection to the database to be backed up by executing the following command at the host (as the database instance user):

```bash
CONNECT TO [DBServer]
```

where:

- `DBServer` — Identifies the database server or the alias of the database to which an application or user is to connect.
For example, to establish a connection to a database named **TEST**, you would execute a command that looks like this:

```
CONNECT TO test
```

4. Temporarily suspend all I/O activity for the database by executing the following command at the host (as the database instance user):

```
SET WRITE SUSPEND FOR DATABASE
```

**IMPORTANT**

At this time, you should not use the `db2_all` command in conjunction with the `SET WRITE SUSPEND FOR DATABASE` command to suspend I/O for a multipartitioned database. Doing so can result in a deadlock situation. If you must suspend I/O for a multipartitioned database, IBM recommends that you start with the highest partition and work your way down to the lowest partition (partition number 0), suspending I/O at each partition.

5. Activate the clone by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] activate -consistent -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

**Note:** The `-consistent` keyword tells Enginuity to use Enginuity Consistency Assist (ECA) to momentarily suspend writes to the source disks while the clone is being activated.

For example, to activate a clone that was created earlier for a device group named **symdb2dg**, execute a command that looks like this:

```
symclone -g symdb2dg activate -consistent -tgt -noprompt
```

And when this command is executed, you should see a message that looks something like this:

'Activate' operation execution is in progress for device group 'symdb2dg'. Please wait...
'Activate' operation successfully executed for device group 'symdb2dg'.

6. If desired, you can verify that data in the source devices used is being copied to the appropriate target devices by executing one of the following commands at the host (as the system administrator or root user):

   `symclone -g [DeviceGroup] verify -copied`

   or

   `symclone -g [DeviceGroup] query`

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to determine whether data is being copied to the set of clone target devices that have been assigned to a device group named `symdb2dg`, execute a command that looks like this:

   `symclone -g symdb2dg query`

When this command is executed, you should see a message that looks something like this:

Device Group (DG) Name: symdb2dg
DG's Type : REGULAR
DG's Symmetrix ID : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td>Logical Sym Tracks</td>
<td>Tracks</td>
<td>Logical Sym Tracks</td>
</tr>
<tr>
<td>Modified</td>
<td>Modified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------</td>
<td>------------</td>
</tr>
<tr>
<td>DEV001 0600</td>
<td>546094</td>
<td>0</td>
<td>TGT001 070F</td>
</tr>
<tr>
<td>DEV002 060A</td>
<td>575960</td>
<td>0</td>
<td>TGT002 0719</td>
</tr>
<tr>
<td>DEV003 0614</td>
<td>572569</td>
<td>0</td>
<td>TGT003 0723</td>
</tr>
<tr>
<td>DEV004 061E</td>
<td>522108</td>
<td>0</td>
<td>TGT004 072D</td>
</tr>
</tbody>
</table>

| Track(s)      | 2216731       | 0     | 0          |
| MB(s)         | 138546         | 0     | 0          |

Legend:

(C): X = The background copy setting is active for this pair.
.. = The background copy setting is not active for this pair.
(G): X = The Target device is associated with a group.
.. = The Target device is not associated with a group.
(D): X = The Clone session is a differential copy session.
Backing Up a DB2 LUW Database Using TimeFinder Technology

(P): X = The pre-copy operation has completed one cycle
. = The pre-copy operation has not completed one cycle

7. Once the clone has been successfully activated, resume all I/O activity for the database by executing the following command at the host (as the database instance user):

```
    SET WRITE RESUME FOR DATABASE
```

When this command is executed, the database is returned to its normal operating state.

**IMPORTANT**

The `SET WRITE RESUME FOR DATABASE` command must be executed from the same connection from which the `SET WRITE SUSPEND FOR DATABASE` command was issued.

8. Terminate the connection to the database established earlier by executing the following command at the host (as the database instance user):

```
    CONNECT RESET
```

After completing these steps, you will have a point-in-time copy of the database (on the clone target devices) that can be used to restore the database if necessary.

As pointed out earlier, databases that have been copied using TimeFinder/Clone are subject to Copy On First Write (COFW) and Copy On Access (COA) penalties when Enginuity 5771 and earlier are used. The COFW penalty arises because the first time a track is written to the source device, if it has not yet been copied to the target device, it must first be copied to the target device before the write from the host will be acknowledged. Subsequent writes to tracks that have already been copied do not suffer this penalty. With Enginuity 5772 and later, COFW is performed asynchronous to the host and has minimal impact on write performance.

The COA penalty arises if a track on a target volume is accessed before it has been copied, requiring it to first be copied from the source volume to the target volume. This causes additional disk read activity to the source volumes and could be a source of disk contention on busy systems. The COA affect can be mitigated by specifying the `-precopy` or `-copy` keyword with the `symclone create` command used.
Creating an online database back up image with TimeFinder/Snap

To create an online back up copy of a DB2 database using TimeFinder/Snap, complete the following steps, in the order shown:

1. Create a relationship between the Symmetrix devices that contain the database to be backed up and the VDEVs and SAVDEVs that are to serve as the target for the virtual device copy and any related data changes by executing the following command at the host (as the system administrator or root user):

   ```bash
   symsnap -g [DeviceGroup] create -svp [SavePool] -noprompt
   ```

   where:
   - **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
   - **SavePool** — Identifies the SAVDEVs that are to be used to store original tracks associated with the virtual device copy that are changed once the copy session is activated.

   For example, to create a copy session relationship for a set of Symmetrix devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

   ```bash
   symsnap -g symdb2dg create -svp DB2_snappool -noprompt
   ```

   And when this command is executed, you should see a message that looks something like this:

   'Create' operation execution is in progress for device group 'symdb2dg'. Please wait...

   'Create' operation successfully executed for device group 'symdb2dg'.

2. Verify that the copy session relationship was created by executing the following command at the host (as the system administrator or root user):

   ```bash
   symsnap list
   ```

   When this command is executed, you should see a message that looks something like this:

   Symmetrix ID: 000190300709
### Backing Up a DB2 LUW Database Using TimeFinder Technology

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>Status</th>
<th>SaveDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sym</td>
<td>Tracks</td>
<td>Sym</td>
<td>G SRC &lt;=</td>
</tr>
<tr>
<td>Protected</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0600</td>
<td>655500</td>
<td>05C0</td>
<td>X Created</td>
</tr>
<tr>
<td>060A</td>
<td>655500</td>
<td>05CA</td>
<td>X Created</td>
</tr>
<tr>
<td>0614</td>
<td>655500</td>
<td>05D4</td>
<td>X Created</td>
</tr>
<tr>
<td>061E</td>
<td>655500</td>
<td>05DE</td>
<td>X Created</td>
</tr>
<tr>
<td>Total</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>Tracks</td>
<td>2949669</td>
<td>MB(s)</td>
<td>184354</td>
</tr>
</tbody>
</table>

Legend:

(G): X = The Target device is associated with a group,
. = The Target device is not associated with a group.

3. Establish a connection to the database to be backed up by executing the following command at the host (as the database instance user):

   ```
   CONNECT TO [DBServer]
   ```

   where:

   - **DBServer** — Identifies the database server or the alias of the database to which an application or user is to connect.

   For example, to establish a connection to a database named **TEST**, you would execute a command that looks like this:

   ```
   CONNECT TO test
   ```

4. Temporarily suspend all I/O activity for the database by executing the following command at the host (as the database instance user):

   ```
   SET WRITE SUSPEND FOR DATABASE
   ```

   **IMPORTANT**

   At this time, you should not use the `db2_all` command in conjunction with the `SET WRITE SUSPEND FOR DATABASE` command to suspend I/O for a multipartitioned database. Doing so can result in a deadlock situation. If you must suspend I/O for a multipartitioned database, IBM recommends that you start with the highest partition and work your way down to the lowest partition (partition number 0), suspending I/O at each partition.
5. Activate the copy session relationship by executing the following command at the host (as the system administrator or root user):

```bash
symsnap -g [DeviceGroup] activate -consistent -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

Note: The `-consistent` keyword tells Enginuity to use Enginuity Consistency Assist (ECA) to momentarily suspend writes to the source disks while the copy session relationship is being activated.

For example, to activate a copy session that was created earlier for a device group named `symdb2dg`, you would execute a command that looks like this:

```bash
symsnap -g symdb2dg activate -consistent -noprompt
```

And when this command is executed, you should see a message that looks something like this:

> 'Activate' operation execution is in progress for device group 'symdb2dg'. Please wait...

> 'Activate' operation successfully executed for device group 'symdb2dg'.

6. If desired, you can verify that the copy session has been activated by executing the following command at the host (as the system administrator or root user):

```bash
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine whether a copy session for the set of Symmetrix devices that have been assigned to a device group named `symdb2dg` has been successfully activated, you would execute a command that looks like this:

```bash
symsnap -g symdb2dg query
```
And if the copy session relationship was successfully activated, you should see a message that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type             : REGULAR
DG's Symmetrix ID     : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
<th>Changed</th>
<th>Protected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
<td>Sym</td>
<td>G</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>DEV001</td>
<td>0600</td>
<td>655499</td>
<td>VDEV001</td>
<td>05C0</td>
<td>X</td>
</tr>
<tr>
<td>DEV002</td>
<td>060A</td>
<td>655499</td>
<td>VDEV002</td>
<td>05CA</td>
<td>X</td>
</tr>
<tr>
<td>DEV003</td>
<td>0614</td>
<td>655499</td>
<td>VDEV003</td>
<td>05D4</td>
<td>X</td>
</tr>
<tr>
<td>DEV004</td>
<td>061E</td>
<td>655499</td>
<td>VDEV001</td>
<td>05DE</td>
<td>X</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Track(s)</td>
<td></td>
<td>1966497</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MB(s)</td>
<td></td>
<td>122906</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Legend:

(G): X = The Target device is associated with this group,
. = The Target device is not associated with this group.

7. Once the activate operation is complete, resume all I/O activity for the database by executing the following command at the host (as the database instance user):

```
SET WRITE RESUME FOR DATABASE
```

When this command is executed, the database is returned to its normal operating state.

**IMPORTANT**

The `SET WRITE RESUME FOR DATABASE` command must be executed from the same connection from which the `SET WRITE SUSPEND FOR DATABASE` command was issued.

8. Terminate the connection to the database established earlier by executing the following command at the host (as the database instance user):

```
CONNECT RESET
```
After completing these steps, you will have a point-in-time copy of the database (in the form of a virtual device copy) that can be used to restore the database if necessary.

As mentioned earlier, it is important to note that databases that have been copied using TimeFinder/Snap are subject to Copy On First Write (COFW) penalties when the copy session is activated if Enginuity 5771 and earlier are used. The COFW penalty arises because the first time a track is written to the source device, if it has not yet been copied to the appropriate SAVDEV, it must first be copied to the SAVDEV before the write from the host will be acknowledged. With Enginuity 5772 and later, COFW is performed asynchronous to the host and has minimal impact on write performance.
Backing up the database copy

Although it is possible to rely solely on EMC’s disk-based replication technology for database back up and recovery, it may be desirable to have a database back up image that was created with the DB2 Backup utility available on separate media. If that is the case, such a back up image can be produced from a disk-based copy of a database that was created using TimeFinder technology; the original database does not have to be the source for a DB2 back up operation.

In order to create a back up image from a disk-based replication copy of a database, the BCV devices, clone target devices, or virtual devices that contain the database copy must be presented to a back up server. (The back up server used can be a separate host or it can be the same host that is used to manage the production database the disk-based replication copy was made from. However, the mount points used must be identical to those of the original database so using the same host can be problematic.) To do this, all LUNs used must be presented and masked, the devices used must be recognized by the host, any logical device groupings used must be imported, any file systems used must be mounted, and the database itself must be cataloged at the server. Once this is done and the back up server has access to the disk-based replication copy of the database, the DB2 Backup utility can be used to create a back up image of the database; the back up image produced can be written to disk, tape, or Tivoli Storage Manager (TSM) – provided the database uses either Automatic Storage (AS) table spaces or Database Managed Space (DMS) table spaces to for data storage. If the database uses System Managed Space (SMS) table spaces to store user data, any attempt to create a back up image using the DB2 Backup utility will fail. (Refer to Chapter 3, “DB2 for Linux, UNIX, and Windows Back Up and Recovery Concepts,” for more information on how to use the DB2 Backup utility.)

It is important to note that when the DB2 Backup utility is used to create a back up image from a disk-based replication copy of a database, the recovery history file for the database that is seen by the back up server will be updated. The recovery history file for the database that the disk-based replication copy was created from will not be modified. Therefore, the original database will not be aware that a back up image has been created for it.
Restoring a damaged or corrupted database typically involves replacing production database files with files that reside in a database copy or back up image and, if appropriate, replaying records stored in transaction log files (to return the database to the state it was in at a specific point in time). The actual recovery process used is determined, in part, by the method used to create the database copy or back up image that is to be utilized for recovery. If a back up image was created with DB2’s back up utility, DB2’s restore utility or recover utility must be used to restore a damaged database. If however, a database copy was made using disk-based replication (for example, EMC TimeFinder technology), similar technology must be used, in conjunction with select DB2 utilities, to facilitate recovery.

This chapter is designed to provide guidelines for using TimeFinder technology to restore DB2 LUW databases residing on EMC Symmetrix arrays that have been backed up with a database copy that was created earlier using TimeFinder/Clone or TimeFinder/Snap. Topics covered include:

◆ Using disk-based replication to restore a DB2 LUW database... 235
◆ Restoring a nonrecoverable DB2 LUW database from an offline database backup copy ................................................................. 237
◆ Restoring a nonrecoverable DB2 LUW database from an online database copy ................................................................. 252
◆ Restoring a recoverable DB2 LUW database from an offline database copy ................................................................. 269
◆ Restoring a recoverable DB2 LUW database from an online database copy ................................................................. 287
Note: The procedures outlined in this chapter make the assumption that all necessary Symmetrix standard devices (STDs), Business Continuance Volume devices (BCVs), clone target devices (TGTs), virtual devices (VDEVs), and save devices (SAVDEVs) have been created and added to a device group. It is also assumed that the appropriate devices contain a copy of the DB2 LUW database that is to be restored.
Using disk-based replication to restore a DB2 LUW database

The primary reason for creating a database back up is to have a copy of mission-critical data available in the event a production database becomes damaged or corrupted. DB2 databases can become damaged or corrupted when any of the following occurs at the server where the DB2 instance is running or at the storage system where the database resides:

- A power failure
- A serious operating system error
- A serious application error
- A hardware failure (for example, memory corruption, disk drive failure, CPU failure, or network failure)
- An “unfenced” stored procedure or user-defined function error

If a back up image exists, a damaged database can be returned to the state it was in at the time the back up image was created should a recovery operation be required.

In Chapter 5, “Backing Up a DB2 LUW Database Using TimeFinder Technology,” we saw that EMC disk-based replication can be used to copy a DB2 LUW database and that the copy produced, in turn, can act as a database back up image that can be used to restore the database in the event it becomes damaged or corrupted. One of the easiest ways to create a disk-based replication copy of a DB2 LUW database is by using the EMC local replication product, TimeFinder. The TimeFinder family of products consists of TimeFinder/Clone and TimeFinder/Snap; both products can be used to create a database copy that can serve as a back up image. The actual process used is determined by the form of TimeFinder chosen and the state the database is in at the time the back up copy is made.

The process used to create a back up copy of a database determines, in part, the steps that must be performed to restore the database from the back up copy produced. Another factor that plays an important part in recovery is whether the database is considered recoverable or nonrecoverable. (Whether a database is considered recoverable or nonrecoverable is determined by the values assigned to the database’s logarchmeth1 and logarchmeth2 configuration parameters; when both of these configuration parameters are assigned the value OFF—which is the default—circular logging is used, and the database is considered nonrecoverable. If either of these parameters is set to a
value other than OFF, archival logging is used and the database is considered recoverable. For more information, refer to “Recoverable and nonrecoverable databases” on page 144.)

IMPORTANT
To perform a roll-forward recovery operation on a recoverable database that has been restored using disk-based replication, storage for the database must be divided into two parts: one for the database and its associated data objects and the other for the database's transaction log files. If both the data objects and the recovery objects reside on the same Symmetrix volume, both the transaction log files and the database's data files will be returned to the state they were in at the time the back up copy was made if a disk-based replication restore operation is performed. As a result, roll-forward recovery will no longer be possible since every transaction log record produced after the database copy was made will be erased by the restore process. By separating data objects from recovery objects, a database can be restored without impacting the transaction log files.

Restoring a database from a back up copy that was made using TimeFinder technology typically involves copying duplicated files back to their original location. However, before such an operation can be performed, the source database must be shut down and all file systems that are used by the database must be unmounted. (Obviously, once the restore operation is complete, any file systems that were unmounted must be remounted before the restored database can be accessed.)

To facilitate recovery, DB2 treats disk-based replication copies of a database as split mirror back up images. (A split mirror is an “instantaneous” copy of a database that is made by mirroring the disk or disks that contain the database's data and splitting the mirror to create a back up copy of the database.) Because of this, before a database that has been restored using disk-based replication can be accessed, it may need to be initialized before it can be used. Such an initialization is performed by executing the DB2 command db2inidb.
Restoring a nonrecoverable DB2 LUW database from an offline database backup copy

In Chapter 5, “Backing Up a DB2 LUW Database Using TimeFinder Technology,” we saw that ideally, a copy of a DB2 LUW database should be taken while the database is offline. Not only does this help ensure that the database is in a consistent state at the time the backup copy is made, but it also greatly simplifies the recovery process—particularly if the database is a nonrecoverable database.

The following sections describe how to restore a nonrecoverable database from a disk-based replication copy that was made using EMC’s TimeFinder technology while the database was offline. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.

Note: When restoring a nonrecoverable database from an offline database backup copy, both the Symmetrix Logical Volume (SLV) devices that contain database data and the SLV devices that contain the transaction logs need to be restored.

Restoring a nonrecoverable database from an offline database back up image that was created with TimeFinder/Clone

If TimeFinder/Clone was used to create a back up copy of a DB2 database, the TimeFinder/Clone restore process can be used to return the database to the state it was in at the time the back up copy was made. TimeFinder/Clone allows a target clone image to be restored back to either the original source device or to an unrelated target device. Prior to Solutions Enabler 6.0, data could be restored from a clone target device back to its source device only by performing a reverse clone operation. The clone relationship was terminated between the source and the target; the target was then used as the source for creating and activating a new clone relationship. With SYMCLI 6.0 and Enginuity 5671 code, a restore can be performed without terminating the original clone session. However, to take advantage of this feature, the clone must have been created with the -differential option specified.
To restore a nonrecoverable DB2 database from an offline back up copy that was created using TimeFinder/Clone, complete the following steps, in the order shown:

1. Verify that the clone target devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Active” and “Copied” state) by executing the following command at the host (as the system administrator or root user):

   \texttt{symclone -g [DeviceGroup] query}

   where:

   - \texttt{DeviceGroup} — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

   For example, to determine if the clone target devices that have been assigned to a device group named \texttt{symdb2dg} are in a valid state for recovery, you would execute a command that looks like this:

   \texttt{symclone -g symdb2dg query}

   And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

<table>
<thead>
<tr>
<th>Device Group (DG) Name: symdb2dg</th>
<th>DG's Type : REGULAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG's Symmetrix ID : 000190300709</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Sym Tracks</td>
<td>Logical Sym Tracks</td>
<td>Modified</td>
<td>CGDP SRC &lt;=</td>
</tr>
<tr>
<td>Protected</td>
<td>Modified</td>
<td>Source Device</td>
<td>Target Device</td>
</tr>
<tr>
<td>0 0 0 tgt719</td>
<td>0 0 0 tgt719</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 tgt70f</td>
<td>0 0 0 tgt70f</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 tgt723</td>
<td>0 0 0 tgt723</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 0 tgt72d</td>
<td>0 0 0 tgt72d</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

   Total | Track(s) | MB(s) |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0.0 0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

   Legend:

   (C): X = The background copy setting is active for this pair.
   (G): X = The Target device is associated with this group.

   . = The Target device is not associated with this group.
2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

```
DEACTIVATE DATABASE [DBAlias]
```

where:

- **DBAlias** — Identifies the alias of the database to be deactivated (stopped).

or

```
FORCE APPLICATION ALL
```

or

```
db2stop FORCE
```

For multipartition databases, the `DEACTIVATE DATABASE` command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

⚠️ **CAUTION**

The `FORCE APPLICATION ALL` command will terminate all connections to all databases under the current DB2 Database Manager instance’s control; the `db2stop FORCE` command will terminate all connections to all databases under the current instance’s control and then stop the instance. (If the `db2stop FORCE` command is issued and applications do not respond within the time frame specified by the `start_stop_time` DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance’s control.

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.
For example, to unmount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```
for i in 1 2 3 4; do umount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

```
df
```

The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.

For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Clone restore process by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] restore -tgt -noprompt
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.
For example, to restore a set of SLV devices that have been assigned to a device group named `symdb2dg` from their corresponding clone target devices, you would execute a command that looks like this:

```
symclone -g symdb2dg restore -tgt -noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `symdb2dg` will be overlaid with data stored on the associated clone target devices.

6. Verify that the TimeFinder/Clone restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] verify -restored
```

or

```
symclone -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symclone -g symdb2dg verify -restored
```

And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

```
All devices in the group 'symdb2dg' are in 'Restored' state.
```

Once the TimeFinder/Clone restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the clone target devices to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. If a version of Solutions Enabler earlier than 6.0 is being used, terminate the clone target/SLV device relationship by executing the following command at the host (as the system administrator or root user):

```
symclone -g symdb2dg terminate -tgt -noprompt
```
Restoring a DB2 LUW Database Using TimeFinder Technology

\[
\text{symclone} \ -g \ [\text{DeviceGroup}] \ \text{terminate} \ -tgt \ -noprompt
\]

where:

- \text{DeviceGroup} — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

However, if Solutions Enabler 6.0 and Enginuity 5671 or later are being used, the clone target/SLV device relationship can be sustained by executing the following command at the host (as the system administrator or root user) instead:

\[
\text{symclone} \ -g \ [\text{DeviceGroup}] \ \text{split} \ -tgt \ -noprompt
\]

where:

- \text{DeviceGroup} — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to terminate the clone target/SLV device relationship between devices that have been assigned to a device group named \text{symdb2dg}, you would execute a command that looks like this:

\[
\text{symclone} \ -g \ \text{symdb2dg} \ \text{terminate} \ -tgt \ -noprompt
\]

And when this command is executed, you should see a message that looks like this:

'Terminate' operation execution is in progress for device group 'symdb2dg'. Please wait...

'Terminate' operation successfully executed for device group 'symdb2dg'.

On the other hand, to split a set of clone target devices that have been assigned to a device group named \text{symdb2dg} from their corresponding SLV devices, you would execute a command that looks more like this:

\[
\text{symclone} \ -g \ \text{symdb2dg} \ \text{split} \ -tgt \ -noprompt
\]

When this command is executed, you should see a message that looks something like this:

'Clone Split' operation execution is in progress for device group 'symdb2dg'. Please wait...

'Clone Split' operation successfully initiated for device group 'symdb2dg'.
8. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
importvg -y 'db2vg1' hdiskpower8
importvg -y 'db2vg2' hdiskpower9
importvg -y 'db2vg3' hdiskpower10
importvg -y 'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```bash
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

9. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

```bash
df
```

The file systems used by the database should now appear in the list of active file systems returned.

After completing these steps, your database recovery is complete. If you did not terminate the clone target/SLV device relationship, you will still have a point-in-time copy of the database available on the
clone target devices used should you need to restore the database again. At this point, you should connect to the database and verify that its contents are as expected.

When the copy of the database stored on the clone target device is no longer needed, it can be deleted by terminating the clone target device/SLV device relationship; this is done by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] terminate -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

---

**Restoring a nonrecoverable database from an offline database back up image that was created with TimeFinder/Snap**

As with TimeFinder/Clone, if TimeFinder/Snap was used to create a back up copy of a DB2 database, the TimeFinder/Snap restore process can be used to return the database to the state it was in at the time the back up copy was made. Prior to Solutions Enabler 5.4, when data was restored from a virtual device back to its source device any other copy sessions that had been created were terminated. Beginning with version 5.4, TimeFinder/Snap restore operations can be performed without affecting the relationship between the source device and any other virtual device copies; additional copy sessions are persistent through a restore operation to the source device.

To restore a nonrecoverable DB2 database from an offline back up copy that was created using TimeFinder/Snap, complete the following steps, in the order shown:

1. Verify that the virtual devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

   ```
symsnap -g [DeviceGroup] query
   ```

   where:

   - **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
For example, to determine if the virtual devices that have been assigned to a device group named symdb2dg are in a valid state for recovery, you would execute a command that looks like this:

`symsnap -g symdb2dg query`

And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type : REGULAR
DG's Symmetrix ID : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>DEV001</td>
<td>0600</td>
<td>655500</td>
<td>vtgt5c0</td>
</tr>
<tr>
<td>DEV002</td>
<td>060A</td>
<td>655500</td>
<td>vtgt5ca</td>
</tr>
<tr>
<td>DEV003</td>
<td>0614</td>
<td>655500</td>
<td>vtgt5d4</td>
</tr>
<tr>
<td>DEV004</td>
<td>061E</td>
<td>655500</td>
<td>vtgt5de</td>
</tr>
</tbody>
</table>

Total: Track(s) 2622000, MB(s) 163875
```

Legend:

(G): X = The Target device is associated with this group,
. = The Target device is not associated with this group.

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

```
DEACTIVATE DATABASE [DBAlias]
```

where:

- `DBAlias` — Identifies the alias of the database to be deactivated (stopped).

or

```
FORCE APPLICATION ALL
```
Restoring a DB2 LUW Database Using TimeFinder Technology

or

\texttt{db2stop FORCE}

For multipartition databases, the \texttt{DEACTIVATE DATABASE} command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

\textbf{CAUTION}

The \texttt{FORCE APPLICATION ALL} command will terminate all connections to all databases under the current DB2 Database Manager instance's control; the \texttt{db2stop FORCE} command will terminate all connections to all databases under the current instance's control and then stop the instance. (If the \texttt{db2stop FORCE} command is issued and applications do not respond within the time frame specified by the \texttt{start_stop_time} DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance's control.

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

For example, to unmount four file systems named \texttt{db2fs1}, \texttt{db2fs2}, \texttt{db2fs3}, and \texttt{db2fs4} on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

\begin{verbatim}
for i in 1 2 3 4; do umount /db2fs$i; done
\end{verbatim}

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

\begin{verbatim}
df
\end{verbatim}

The file systems should no longer appear in the list of active file systems returned.
4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.

For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```plaintext
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```plaintext
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Snap restore process by executing the following command at the host (as the system administrator or root user):

```plaintext
symsnap -g [DeviceGroup] restore –noprompt
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to restore a set of SLV devices that have been assigned to a device group named `symdb2dg` from their corresponding virtual devices, you would execute a command that looks like this:

```plaintext
symsnap -g symdb2dg restore –noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `symdb2dg` will be overlaid with data stored on the associated virtual devices.

6. Verify that the TimeFinder/Snap restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

```plaintext
symsnap -g [DeviceGroup] verify -restored
```

or

```plaintext
symsnap -g [DeviceGroup] query
```
where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symsnap -g symdb2dg verify -restored
```

And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

```
All devices in the group 'symdb2dg' are in 'Restored' state.
```

Once the TimeFinder/Snap restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately; while data is copied from SAVDEV devices to the appropriate SLV devices any tracks that are accessed that have not yet been copied will be copied on demand.

7. When the TimeFinder/Snap restore process is initiated, both the virtual devices used and the corresponding SLV devices are set to a "Not-Ready" state (that is, they are offline to host activity). Once the restore operation commences, all source devices are placed in a "Ready" state. When the restore process is complete, terminate the existing internal copy session between the source and virtual/save devices by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -restored -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to terminate the internal copy session that has been established for devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symsnap -g symdb2dg terminate -restored -noprompt
```

And when this command is executed, you should see a message that looks something like this:
'Terminate' operation execution is in progress for
device group 'symdb2dg'. Please wait...

'Terminate' operation successfully executed for device
group 'symdb2dg'.

8. Verify that the virtual devices that contain the point-in-time copy
of the database have been returned to a valid state for recovery
(the “Associated” and “CopyOnWrite” state) by executing the
following command at the host (as the system administrator or
root user):

```
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that
  TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been
assigned to a device group named `symdb2dg` have been returned
to a valid state for recovery, you would execute a command that
looks like this:

```
symsnap -g symdb2dg query
```

And when this command is executed, you should see a message
that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type             : REGULAR
DG's Symmetrix ID     : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Logical</th>
<th>Sym</th>
<th>Tracks</th>
<th>Target Device</th>
<th>Logical</th>
<th>Sym</th>
<th>Tracks</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Changed</td>
<td>SRC &lt;=&gt; TGT (%)</td>
</tr>
<tr>
<td>DEV001</td>
<td>0600</td>
<td>0600</td>
<td>655500</td>
<td>vtgt5c0</td>
<td>05C0</td>
<td></td>
<td></td>
<td>0</td>
<td>CopyOnWrite 0</td>
</tr>
<tr>
<td>DEV002</td>
<td>060A</td>
<td>060A</td>
<td>655500</td>
<td>vtgt5ca</td>
<td>05CA</td>
<td></td>
<td></td>
<td>0</td>
<td>CopyOnWrite 0</td>
</tr>
<tr>
<td>DEV003</td>
<td>0614</td>
<td>0614</td>
<td>655500</td>
<td>vtgt5d4</td>
<td>05D4</td>
<td></td>
<td></td>
<td>0</td>
<td>CopyOnWrite 0</td>
</tr>
<tr>
<td>DEV004</td>
<td>061E</td>
<td>061E</td>
<td>655500</td>
<td>vtgt5de</td>
<td>05DE</td>
<td></td>
<td></td>
<td>0</td>
<td>CopyOnWrite 0</td>
</tr>
</tbody>
</table>

Total
Track(s) 2622000 0
MB(s) 163875 0

Legend:
9. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
importvg -y 'db2vg1' hdiskpower8
importvg -y 'db2vg2' hdiskpower9
importvg -y 'db2vg3' hdiskpower10
importvg -y 'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```bash
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

10. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

```bash
df
```

The file systems used by the database should now appear in the list of active file systems returned.
After completing these steps, your database recovery is complete. At this point, you should connect to the database and verify that its contents are as expected. If the virtual device copy used is needed for further processing after the TimeFinder/Snap restore process is complete, the corresponding virtual devices must be returned to a “Ready” state. This can be accomplished by executing a command that looks something like this:

```bash
symld -g [DeviceGroup] ready [VirtualDeviceID] -vdev
```

where:
- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
- **VirtualDeviceID** — Identifies a virtual device, by ID, that is to be returned to a “Ready” state.

Alternately, if after the restore process completes, the virtual device/SLV device relationship is no longer needed, it can be terminated by executing the following command at the host (as the system administrator or root user):

```bash
symsnap -g [DeviceGroup] terminate -noprompt
```

where:
- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
Restoring a nonrecoverable DB2 LUW database from an online database copy

In environments where it is not always possible to take a DB2 LUW database offline in order to back it up, an alternative approach is to make a copy of the database while it remains online. As noted in Chapter 5, “Back Up a DB2 LUW Database Using TimeFinder Technology,” an online back up copy of a DB2 database can be created by temporarily suspending all database I/O, duplicating the database’s data files, and then resuming I/O.

The following sections describe how to restore a nonrecoverable database from a disk-based replication copy that was made using EMC TimeFinder technology while the database was online. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.

Note: When restoring a nonrecoverable database from an online database backup copy, both the SLV devices that contain database data and the SLV devices that contain the transaction logs need to be restored.

Restoring a nonrecoverable database from an online database back up image that was created with TimeFinder/Clone

If TimeFinder/Clone was used to create a back up copy of a DB2 database, the TimeFinder/Clone restore process can be used to return the database to the state it was in at the time the back up copy was made. TimeFinder/Clone allows a target clone image to be restored back to the original source device or to an unrelated target device. Prior to Solutions Enabler 6.0, data could be restored from a clone target device back to its source device only by performing a reverse clone operation. The clone relationship was terminated between the source and the target; the target was then used as the source for creating and activating a new clone relationship. With SYMCLI 6.0 and Enginuity 5671 code, a restore can be performed without terminating the original clone session. However, to take advantage of this feature, the clone must have been created with the **-differential** option specified.
To restore a nonrecoverable DB2 database from an online back up copy that was created using TimeFinder/Clone, complete the following steps, in the order shown:

1. Verify that the clone target devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Active” and “Copied” state) by executing the following command at the host (as the system administrator or root user):

   ```
   symclone -g [DeviceGroup] query
   ```

   where:
   - **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

   For example, to determine if the clone target devices that have been assigned to a device group named `symdb2dg` are in a valid state for recovery, you would execute a command that looks like this:

   ```
   symclone -g symdb2dg query
   ```

   And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

   ```
   Device Group (DG) Name: symdb2dg
   DG's Type             : REGULAR
   DG's Symmetrix ID     : 000190300709

   Source Device                   Target Device            State     Copy
   --------------------------------- ---------------------------- ------------ ----
   Logical   Sym  Tracks    Tracks   Logical   Sym  Tracks   CGDP SRC <=> TGT  (%)
   --------------------------------- ---------------------------- ------------ ----
   DEV001    0600         0        0 tgt719    0719        0 XX.. Copied       100
   DEV002    060A         0        0 tgt70f    070F        0 XX.. Copied       100
   DEV003    0614         0        0 tgt723    0723        0 XX.. Copied       100
   DEV004    061E         0        0 tgt72d    072D        0 XX.. Copied       100
   Total           -------- --------                --------
   Track(s)             0        0                       0
   MB(s)                0.0      0.0                     0.0

   Legend:
   - (C): X = The background copy setting is active for this pair.
   - . = The background copy setting is not active for this pair.
   - (G): X = The Target device is associated with this group.
   - . = The Target device is not associated with this group.
   ```
2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

DEACTIVATE DATABASE [DBAlias]

where:

- **DBAlias** — Identifies the alias of the database to be deactivated (stopped).

or

FORCE APPLICATION ALL

or

db2stop FORCE

For multipartition databases, the **DEACTIVATE DATABASE** command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

**CAUTION**

The **FORCE APPLICATION ALL** command will terminate all connections to all databases under the current DB2 Database Manager instance’s control; the **db2stop FORCE** command will terminate all connections to all databases under the current instance’s control and then stop the instance. (If the **db2stop FORCE** command is issued and applications do not respond within the time frame specified by the **start_stop_time** DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance’s control.
3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

For example, to unmount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do umount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

```bash
df
```

The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.

For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Clone restore process by executing the following command at the host (as the system administrator or root user):

```bash
symclone -g [DeviceGroup] restore -tgt -noprompt
```

where:
Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

Restoring a DB2 LUW Database Using TimeFinder Technology

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on. For example, to restore a set of SLV devices that have been assigned to a device group named `symdb2dg` from their corresponding clone target devices, you would execute a command that looks like this:

```
symclone -g symdb2dg restore -tgt -noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `symdb2dg` will be overlaid with data stored on the associated clone target devices.

6. Verify that the TimeFinder/Clone restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] verify -restored
```

or

```
symclone -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symclone -g symdb2dg verify -restored
```

And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

```
All devices in the group 'symdb2dg' are in 'Restored' state.
```

Once the TimeFinder/Clone restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the clone target devices to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.
7. If a version of Solutions Enabler earlier than 6.0 is being used, terminate the clone target/SLV device relationship by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] terminate -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

However, if Solutions Enabler 6.0 and Enginuity 5671 or later are being used, the clone target/SLV device relationship can be sustained by executing the following command at the host (as the system administrator or root user) instead:

```
symclone -g [DeviceGroup] split -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to terminate the clone target/SLV device relationship between devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symclone -g symdb2dg terminate -tgt -noprompt
```

And when this command is executed, you should see a message that looks like this:

>'Terminate' operation execution is in progress for device group 'symdb2dg'. Please wait...

>'Terminate' operation successfully executed for device group 'symdb2dg'.

On the other hand, to split a set of clone target devices that have been assigned to a device group named `symdb2dg` from their corresponding SLV devices, you would execute a command that looks more like this:

```
symclone -g symdb2dg split -tgt -noprompt
```

When this command is executed, you should see a message that looks something like this:
'Clone Split' operation execution is in progress for 
device group 'symdb2dg'. Please wait...

'Clone Split' operation successfully initiated for 
device group 'symdb2dg'.

8. Activate all SLV-related volume groups used, as well as their 
associated logical volumes. The actual command used to activate 
a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, 
`db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would 
execute a set of commands that look something like this (as the 
root user):

```bash
importvg -y 'db2vg1' hdiskpower8
importvg -y 'db2vg2' hdiskpower9
importvg -y 'db2vg3' hdiskpower10
importvg -y 'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a 
Linux server, you might execute a set of commands that look 
more like this (as the root user):

```bash
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

9. Once the appropriate volume groups have been activated, 
remount all file systems that are used by the database. The actual 
command used to mount a file system is operating system 
dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, 
`db2fs3`, and `db2fs4` on an AIX or Linux server, you would 
execute a command that looks something like this (as the root 
user):

```bash
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file 
systems have been mounted by executing a command that looks 
like this:

```bash
df
```
The file systems used by the database should now appear in the list of active file systems returned.

10. At this point, the database has been restored, but because the backup copy used to restore the database was created while all database I/O was suspended, the restored database is currently in the “Write-Suspended” state. Take the database out of “Write-Suspended” state by executing the following command at the host (as the database instance user):

```
RESTART DATABASE [DBName] WRITE RESUME
```

where:

- **DBName** — Identifies the name or alias of the database to be restarted.

For multipartition databases, the `RESTART DATABASE` command must be executed on all partitions that belong to the database specified. This can be accomplished by executing the following command instead:

```
db2_all || “db2 RESTART DATABASE [DBName] WRITE RESUME”
```

Note: If the command `db2stop FORCE` was used to terminate all connections to the database, the command `db2start` must be executed before the `RESTART DATABASE` command will execute successfully.

For example, to take a nonrecoverable database named TEST, out of “Write-Suspended” state, you would execute a command that looks like this:

```
RESTART DATABASE test WRITE RESUME
```

After completing these steps, your database recovery is complete. If you did not terminate the clone target/SLV device relationship, you will still have a point-in-time copy of the database available on the clone target devices used should you need to restore the database again. At this point, you should connect to the database and verify that the database is as expected.

When the copy of the database stored on the clone target device is no longer needed, it can be deleted by terminating the clone target device/SLV device relationship; this is done by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] terminate -tgt -noprompt
```
Restoring a nonrecoverable database from an online database back up image that was created with TimeFinder/Snap

As with TimeFinder/Clone, if TimeFinder/Snap was used to create a back up copy of a DB2 database, the TimeFinder/Snap restore process can be used to return the database to the state it was in at the time the back up copy was made. Prior to Solutions Enabler 5.4, when data was restored from a virtual device back to its source device any other copy sessions that had been created were terminated. Beginning with version 5.4, TimeFinder/Snap restore operations can be performed without affecting the relationship between the source device and any other virtual device copies; additional copy sessions are persistent through a restore operation to the source device.

To restore a nonrecoverable DB2 database from an online back up copy that was created using TimeFinder/Snap, complete the following steps, in the order shown:

1. Verify that the virtual devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

   ```bash
   symsnap -g [DeviceGroup] query
   ```

   where:

   - **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

   For example, to determine if the virtual devices that have been assigned to a device group named `symdb2dg` are in a valid state for recovery, you would execute a command that looks like this:

   ```bash
   symsnap -g symdb2dg query
   ```

   And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:
Restoring a nonrecoverable DB2 LUW database from an online database copy

Device Group (DG) Name: symdb2dg
DG's Type : REGULAR
DG's Symmetrix ID : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td>Changed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>DEV001 0600</td>
<td>655500 vtgt5c0 05C0</td>
<td>X</td>
<td>19</td>
</tr>
<tr>
<td>DEV002 060A</td>
<td>655500 vtgt5ca 05CA</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>DEV003 0614</td>
<td>655500 vtgt5d4 05D4</td>
<td>X</td>
<td>7</td>
</tr>
<tr>
<td>DEV004 061E</td>
<td>655500 vtgt5de 05DE</td>
<td>X</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>2622000</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>MB(s)</td>
<td>163875</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Legend:

(G): X = The Target device is associated with this group,
     . = The Target device is not associated with this group.

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

   DEACTIVATE DATABASE [DBAlias]

   where:

   - DBAlias — Identifies the alias of the database to be deactivated (stopped).

   or

   FORCE APPLICATION ALL

   or

   db2stop FORCE

For multipartition databases, the DEACTIVATE DATABASE command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.
Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent. For example, to unmount four file systems named db2fs1, db2fs2, db2fs3, and db2fs4 on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

   for i in 1 2 3 4; do umount /db2fs$i; done

   When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

   df

   The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent. For example, to deactivate four volume groups named db2vg1, db2vg2, db2vg3, and db2vg4 on an AIX server, you would execute a set of commands that look something like this (as the root user):

   for i in 1 2 3 4; do varyoffvg db2vg$i; done
   for i in 1 2 3 4; do vgexport db2vg$i; done
On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Snap restore process by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] restore -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to restore a set of SLV devices that have been assigned to a device group named `symdb2dg` from their corresponding virtual devices, you would execute a command that looks like this:

```
symsnap -g symdb2dg restore -noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `symdb2dg` will be overlaid with data stored on the associated virtual devices.

6. Verify that the TimeFinder/Snap restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] verify -restored
```

or

```
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

```
symsnap -g symdb2dg verify -restored
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```
And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

All devices in the group 'symdb2dg' are in 'Restored' state.

Once the TimeFinder/Snap restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the SAVDEV devices used to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. When the TimeFinder/Snap restore process is initiated, both the VDEV devices and the corresponding SLV devices are set to a "Not-Ready" state (that is, they are offline to host activity). Once the restore operation commences, all source devices are placed in a "Ready" state. When the restore process is complete, terminate the existing internal copy session between the source and virtual/save devices by executing the following command at the host (as the system administrator or root user):

`symsnap -g [DeviceGroup] terminate -restored -noprompt`

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to terminate the internal copy session that has been established for devices that have been assigned to a device group named `symdb2dg`, you would execute a command that looks like this:

`symsnap -g symdb2dg terminate -restored -noprompt`

And when this command is executed, you should see a message that looks something like this:

'Terminate' operation execution is in progress for device group 'symdb2dg'. Please wait...

'Terminate' operation successfully executed for device group 'symdb2dg'.
8. Verify that the virtual devices that contain the point-in-time copy of the database have been returned to a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been assigned to a device group named `symdb2dg` have been returned to a valid state for recovery, you would execute a command that looks like this:

```
symsnap -g symdb2dg query
```

And when this command is executed, you should see a message that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type : REGULAR
DG's Symmetrix ID : 000190300709

| Source Device | Logical | Sym | Tracks | Target Device | Logical | Sym | G | Changed | Tracks | SRC <=> TGT (%)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| DEVO01        | 0600    | 655500 | vtgt5c0   | 05C0 | X          | 0 CopyOnWrite | 0
| DEVO02        | 060A    | 655500 | vtgt5ca   | 05CA | X          | 0 CopyOnWrite | 0
| DEVO03        | 0614    | 655500 | vtgt5d4   | 05D4 | X          | 0 CopyOnWrite | 0
| DEVO04        | 061E    | 655500 | vtgt5de   | 05DE | X          | 0 CopyOnWrite | 0
| Total         | --------|-------|---------|---------------|---------|-----|---|---------|--------|----------------|
| Track(s)      | 2622000 |      |         |               |         |     |   |         |        |                |
| MB(s)         | 163875  |      |         |               |         |     |   |         |        |                |

Legend:

(G): X = The Target device is associated with this group,
. = The Target device is not associated with this group.

Restoring a nonrecoverable DB2 LUW database from an online database copy
9. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```
importvg -y'db2vg1' hdiskpower8
importvg -y'db2vg2' hdiskpower9
importvg -y'db2vg3' hdiskpower10
importvg -y'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

10. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

```
df
```

The file systems used by the database should now appear in the list of active file systems returned.

11. At this point, the database has been restored, but because the backup copy used to restore the database was created while all database I/O was suspended, the restored database is currently
in the “Write-Suspended” state. Take the database out of
“Write-Suspended” state by executing the following command at
the host (as the database instance user):

```
RESTART DATABASE [DBName] WRITE RESUME
```

where:

- **DBName** — Identifies the name or alias of the database to be
  restarted.

For multipartition databases, the **RESTART DATABASE** command
must be executed on all partitions that belong to the database
specified. This can be accomplished by executing the following
command instead:

```
db2_all || "db2 RESTART DATABASE [DBName] WRITE RESUME"
```

Note: If the command **db2stop FORCE** was used to terminate all
connections to the database, the command **db2start** must be executed
before the **RESTART DATABASE** command will execute successfully.

For example, to take a nonrecoverable database named **TEST** out
of “Write-Suspended” state, you would execute a command that
looks like this:

```
RESTART DATABASE test WRITE RESUME
```

After completing these steps, your database recovery is complete. At
this point, you should connect to the database and verify that its
contents are as expected. If the virtual device copy used is needed for
further processing after the TimeFinder/Snap restore process is
complete, the corresponding virtual devices must be returned to a
“Ready” state. This can be accomplished by executing a command
that looks something like this:

```
symld -g [DeviceGroup] ready [VirtualDeviceID] -vdev
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that
  TimeFinder/Snap control operations are to be performed on.
- **VirtualDeviceID** — Identifies the a virtual device, by ID, that is to
  be returned to a “Ready” state.
Restoring a DB2 LUW Database Using TimeFinder Technology

Alternately, if after the restore process completes, the virtual device/SLV device relationship is no longer needed, it can be terminated by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -noprompt
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
The following sections describe how to restore a recoverable database from a disk-based replication copy that was made using EMC TimeFinder technology while the database was offline. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.

Note: When restoring a recoverable database from an offline database backup copy, just the SLV devices that contain database data should be restored; the SLV devices that contain the transaction logs should be left as they are so that the logs can be used for roll-forward recovery.

Restoring a recoverable database from an offline database backup copy that was created with TimeFinder/Clone

If TimeFinder/Clone was used to create a back up copy of a DB2 database, the TimeFinder/Clone restore process can be used to return the database to the state it was in at the time the back up copy was made. To restore a recoverable DB2 database from an offline back up copy that was created using TimeFinder/Clone, complete the following steps, in the order shown:

1. Verify that the clone target devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Active” and “Copied” state) by executing the following command at the host (as the system administrator or root user):

   `symclone -g [DeviceGroup] query`

   where:

   • DeviceGroup — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to determine if the clone target devices that have been assigned to a device group named symdb2dg are in a valid state for recovery, you would execute a command that looks like this:

   `symclone -g symdb2dg query`
And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

Device Group (DG) Name: symdb2dg
DG's Type: REGULAR
DG's Symmetrix ID: 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protected</td>
<td>Modified</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>Dev001</td>
<td>0600</td>
<td>0</td>
<td>tgt719</td>
</tr>
<tr>
<td>Dev002</td>
<td>060A</td>
<td>0</td>
<td>tgt70f</td>
</tr>
<tr>
<td>Dev003</td>
<td>0614</td>
<td>0</td>
<td>tgt723</td>
</tr>
<tr>
<td>Dev004</td>
<td>061E</td>
<td>0</td>
<td>tgt72d</td>
</tr>
</tbody>
</table>

Total: 0 0 0 0 0 0 0 0 0 0 0 0 0

Legend:

(C): X = The background copy setting is active for this pair.
(G): X = The Target device is associated with this group.
(D): X = The Clone session is a differential copy session.
(P): X = The pre-copy operation has completed one cycle

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

```sql
DEACTIVATE DATABASE [DBAlias]
```

where:

- `DBAlias` — Identifies the alias of the database to be deactivated (stopped).

or

```sql
FORCE APPLICATION ALL
```

or

```sql
db2stop FORCE
```
For multipartition databases, the **DEACTIVATE DATABASE** command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

---

**CAUTION**

The **FORCE APPLICATION ALL** command will terminate all connections to all databases under the current DB2 Database Manager instance's control; the **db2stop FORCE** command will terminate all connections to all databases under the current instance's control and then stop the instance. (If the **db2stop FORCE** command is issued and applications do not respond within the time frame specified by the **start_stop_time** DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance's control.

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

For example, to unmount four file systems named db2fs1, db2fs2, db2fs3, and db2fs4 on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do umount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

```bash
df
```

The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.
For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Clone restore process by executing the following command at the host (as the system administrator or root user):

```bash
symclone -g [DeviceGroup] restore -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to restore a set of SLV devices that have been assigned to a device group named `db2datadg` from their corresponding clone target devices, you would execute a command that looks like this:

```bash
symclone -g db2datadg restore -tgt -noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `db2datadg` will be overlaid with data stored on the associated clone target devices.

**WARNING**

Do not initiate a TimeFinder/Clone restore operation on volume(s) that contain transaction log files; if such an operation is performed, roll-forward recovery will no longer be possible - all transaction log records produced since the database was copied will overwritten by the restore operation.

6. Verify that the TimeFinder/Clone restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

```bash
for i in 1 2 3 4; do varyvg vg restore -tgt -noprompt; done
for i in 1 2 3 4; do vgexport vg$i; done
```
Restoring a recoverable DB2 LUW database from an offline database copy

symclone -g [DeviceGroup] verify -restored

or

symclone -g [DeviceGroup] query

where:

- DeviceGroup — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named db2datadg, you would execute a command that looks like this:

symclone -g db2datadg verify -restored

And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

All devices in the group 'db2datadg' are in 'Restored' state.

Once the TimeFinder/Clone restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the clone target devices to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. If a version of Solutions Enabler earlier than 6.0 is being used, terminate the clone target/SLV device relationship by executing the following command at the host (as the system administrator or root user):

symclone -g [DeviceGroup] terminate -tgt -noprompt

where:

- DeviceGroup — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

However, if Solutions Enabler 6.0 and Enginiuity 5671 or later are being used, the clone target/SLV device relationship can be sustained by executing the following command at the host (as the system administrator or root user) instead:

symclone -g [DeviceGroup] split -tgt -noprompt

where:
Restoring a DB2 LUW Database Using TimeFinder Technology

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to terminate the clone target/SLV device relationship between devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:

```
symclone -g db2datadg terminate -tgt -noprompt
```

And when this command is executed, you should see a message that looks like this:

>'Terminate' operation execution is in progress for device group 'db2datadg'. Please wait...

>'Terminate' operation successfully executed for device group 'db2datadg'.

On the other hand, to split a set of clone target devices that have been assigned to a device group named `db2datadg` from their corresponding SLV devices, you would execute a command that looks more like this:

```
symclone -g db2datadg split -tgt -noprompt
```

When this command is executed, you should see a message that looks something like this:

>'Clone Split' operation execution is in progress for device group 'db2datadg'. Please wait...

>'Clone Split' operation successfully initiated for device group 'db2datadg'.

8. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```
importvg -y 'db2vg1' hdiskpower8
importvg -y 'db2vg2' hdiskpower9
importvg -y 'db2vg3' hdiskpower10
importvg -y 'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```
On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```bash
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

9. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

```bash
df
```

The file systems used by the database should now appear in the list of active file systems returned.

10. At this point, the database has been returned to the state it was in at the time the back up copy was made. However, since the database was configured for archival logging and because the transaction logs for the database were not overwritten during the recovery process, it is possible to reapply records stored in the database’s transaction log files to return the database to the state it was in at a point-in-time after the back up copy was made. But, in order to do this, the restored database must first be placed in the "Roll-Forward Pending" state. Place the restored database in "Roll-Forward Pending" state by executing the following command at the host (as the database instance user):

```bash
db2rfpen ON [DBName]
```

where:

- `DBName` — Identifies the name of the database that is to be put in “Roll-Forward Pending” state.
For example, to place a recoverable database named **TEST** in the “Roll-Forward Pending” state, you would execute a command that looks like this:

```
db2rfpen ON test
```

And when this command is executed, you should see a message that looks something like this:

```
__________________________________________________________________
____    D B 2 R F P E N    ____
IBM - Reset ROLLFORWARD Pending State

The db2rfpen tool is a utility to switch on the database rollforward pending state.
It will also reset the database role to STANDARD if the database is identified using the database_alias option.

In a non-HADR environment, this tool should be used under the advisement of DB2 service.
In an HADR environment, this tool can be used to reset the database role to STANDARD.

SYNTAX: db2rfpen on < database_alias | -path log_file_header_path | -file log_file_header >
```

11. Once the database has been put in the “Roll-Forward Pending” state, perform a roll-forward recovery operation by executing one of the following commands at the host (as the database instance user):

```
ROLLFORWARD DATABASE [DBName] TO END OF LOGS AND STOP
```

or

```
ROLLFORWARD DATABASE [DBName] TO [Timestamp] AND STOP
```

where:

- **DBName** — Identifies by name, the database that roll-forward recovery is to be performed on.
Restoring a DB2 LUW Database Using TimeFinder Technology

- **TimeStamp** — Identifies a specific point-in-time to which all committed transactions are to be rolled forward (including the transaction committed precisely at that date and time).

Note: If the command `db2stop FORCE` was used to terminate all connections to the database, the command `db2start` must be executed before the `ROLLFORWARD DATABASE` command will execute successfully.

For example, to perform a roll-forward recovery operation on a recoverable database named TEST, you could execute a command that looks like this:

**ROLLFORWARD DATABASE test TO END OF LOGS AND STOP**

And when this command is executed, you should see a message that looks something like this:

**Rollforward Status**

- Input database alias = TEST
- Number of nodes have returned status = 1
- Node number = 0
- Rollforward status = not pending
- Next log file to be read =
- Log files processed = S0000003.LOG - S0000003.LOG
- Last committed transaction = 2008-12-11-15.00.29.000000 UTC

DB20000I The ROLLFORWARD command completed successfully.

For multipartition databases, the `ROLLFORWARD DATABASE` command must be executed from the catalog partition.

After completing these steps, your database recovery is complete. If you did not terminate the clone target/SLV device relationship, you will still have a point-in-time copy of the database available on the clone target devices used should you need to restore the database again. At this point, you should connect to the database and verify that its contents are as expected.

When the copy of the database stored on the clone target device is no longer needed, it can be deleted by terminating the clone target device/SLV device relationship; this is done by executing the following command at the host (as the system administrator or root user):

`symclone -g [DeviceGroup] terminate -tgt -noprompt`
Restoring a recoverable database from an offline database back up image that was created with TimeFinder/Snap

As with TimeFinder/Clone, if TimeFinder/Snap was used to create a back up copy of a DB2 database, the TimeFinder/Snap restore process can be used to return the database to the state it was in at the time the back up copy was made. To restore a recoverable DB2 database from an offline back up copy that was created using TimeFinder/Snap, complete the following steps, in the order shown:

1. Verify that the virtual devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

```bash
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been assigned to a device group named `symdb2dg` are in a valid state for recovery, you would execute a command that looks like this:

```bash
symsnap -g symdb2dg query
```

And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

```
Device Group (DG) Name: symdb2dg
DG's Type            : REGULAR
DG's Symmetrix ID    : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
</table>
|                 |                 | Protected  | Changed
| Logical Sym Tracks | Logical Sym G Tracks | SRC <=> TGT (%) |
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
Restoring a recoverable DB2 LUW database from an offline database copy

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

   **DEACTIVATE DATABASE [DBAlias]**

   where:

   • **DBAlias** — Identifies the alias of the database to be deactivated (stopped).

   or

   **FORCE APPLICATION ALL**

   or

   **db2stop FORCE**

   For multipartition databases, the **DEACTIVATE DATABASE** command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

   **CAUTION**

   The **FORCE APPLICATION ALL** command will terminate all connections to all databases under the current DB2 Database Manager instance's control; the **db2stop FORCE** command will terminate all connections to all databases under the current instance's control and then stop the instance. (If the **db2stop FORCE** command is issued and applications do not respond within the time frame specified by the **start_stop_time** DB2 Database Manager parameter, the database instance will not stop.)
configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance’s control.

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

For example, to unmount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do umount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

```bash
df
```

The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.

For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```
5. Initiate a TimeFinder/Snap restore process by executing the following command at the host (as the system administrator or root user):

   `sym snap -g [DeviceGroup] restore -noprompt`

   where:

   - `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

   For example, to restore a set of SLV devices that have been assigned to a device group named `db2datadg` from their corresponding virtual devices, you would execute a command that looks like this:

   `sym snap -g db2datadg restore -noprompt`

   When this command is executed, all data stored on the SLV devices in the device group named `db2datadg` will be overlaid with data stored on the associated virtual devices.

   **WARNING**

   *Do not initiate a TimeFinder/Snap restore operation on volume(s) that contain transaction log files; if such an operation is performed, roll-forward recovery will no longer be possible - all transaction log records produced since the database was copied will overwritten by the restore operation.*

6. Verify that the TimeFinder/Snap restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):

   `sym snap -g [DeviceGroup] verify -restored`

   or

   `sym snap -g [DeviceGroup] query`

   where:

   - `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

   For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:

   `sym snap -g db2datadg verify -restored`
And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

All devices in the group 'db2datadg' are in 'Restored' state.

Once the TimeFinder/Snap restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the SAVDEV devices used to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. When the TimeFinder/Snap restore process is initiated, both the VDEV devices and the corresponding SLV devices are set to a "Not-Ready" state (that is, they are offline to host activity). Once the restore operation commences, all source devices are placed in a "Ready" state. When the restore process is complete, terminate the existing internal copy session between the source and virtual/save devices by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -restored -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to terminate the internal copy session that has been established for devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:

```
symsnap -g db2datadg terminate -restored -noprompt
```

And when this command is executed, you should see a message that looks something like this:

'Terminate' operation execution is in progress for device group 'db2datadg'. Please wait...

'Terminate' operation successfully executed for device group 'db2datadg'.
8. Verify that the virtual devices that contain the point-in-time copy of the database have been returned to a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

\[ \text{symsnap} \ -g \ [\text{DeviceGroup}] \ \text{query} \]

where:

- \text{DeviceGroup} — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been assigned to a device group named \textit{db2datadg} have been returned to a valid state for recovery, you would execute a command that looks like this:

\[ \text{symsnap} \ -g \ db2datadg \ \text{query} \]

And when this command is executed, you should see a message that looks something like this:

```
Device Group (DG) Name: db2datadg
DG's Type             : REGULAR
DG's Symmetrix ID     : 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>DEV001</td>
<td>0600</td>
<td>655500</td>
<td>vtgt5c0</td>
</tr>
<tr>
<td>DEV002</td>
<td>060A</td>
<td>655500</td>
<td>vtgt5ca</td>
</tr>
<tr>
<td>DEV003</td>
<td>0614</td>
<td>655500</td>
<td>vtgt5d4</td>
</tr>
</tbody>
</table>

Total

| Track(s)     | 2622000       |
| MB(s)        | 163875        |

Legend:

(G): X = The Target device is associated with this group,
. = The Target device is not associated with this group.

9. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.
For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
importvg -y'db2vg1' hdiskpower8
importvg -y'db2vg2' hdiskpower9
importvg -y'db2vg3' hdiskpower10
importvg -y'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```bash
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

10. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do mount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

```bash
df
```

The file systems used by the database should now appear in the list of active file systems returned.

11. At this point, the database has been returned to the state it was in at the time the back up copy was made. However, since the database was configured for archival logging and because the transaction logs for the database were not overwritten during the recovery process, it is possible to reapply records stored in the database’s transaction log files to return the database to the state it was in at a point-in-time after the back up copy was made. But,
in order to do this, the restored database must first be placed in the “Roll-Forward Pending” state. Place the restored database in “Roll-Forward Pending” state by executing the following command at the host (as the database instance user):

```
db2rfpen ON [DBName]
```

where:

- `DBName` — Identifies the name of the database that is to be put in “Roll-Forward Pending” state.

For example, to place a recoverable database named `TEST` in the “Roll-Forward Pending” state, you would execute a command that looks like this:

```
db2rfpen ON test
```

And when this command is executed, you should see a message that looks something like this:

```
__________________________________________________________________
____ DB2RFPEN ______
IBM - Reset ROLLFORWARD Pending State
The db2rfpen tool is a utility to switch on the database rollforward pending state.
It will also reset the database role to STANDARD if the database is identified using the database_alias option.
In a non-HADR environment, this tool should be used under the advisement of DB2 service.
In an HADR environment, this tool can be used to reset the database role to STANDARD.
SYNTAX: db2rfpen on < database_alias | -path log_file_header_path | -file log_file_header >
__________________________________________________________________
Original Rollforward Pending state is Off.
Setting rollforward pending State to On.
```

12. Once the database has been put in the “Roll-Forward Pending” state, perform a roll-forward recovery operation by executing one of the following commands at the host (as the database instance user):

```
ROLLFORWARD DATABASE [DBName] TO END OF LOGS AND STOP
```
or

ROLLFORWARD DATABASE [DBName] TO [Timestamp] AND STOP

where:

- **DBName** — Identifies the name of the database that roll-forward recovery is to be performed on.
- **TimeStamp** — Identifies a specific point-in-time to which all committed transactions are to be rolled forward (including the transaction committed precisely at that date and time).

Note: If the command `db2stop FORCE` was used to terminate all connections to the database, the command `db2start` must be executed before the `ROLLFORWARD DATABASE` command will execute successfully.

For example, to perform a roll-forward recovery operation on a recoverable database named `TEST`, you could execute a command that looks like this:

ROLLFORWARD DATABASE test TO END OF LOGS AND STOP

And when this command is executed, you should see a message that looks something like this:

```
Rollforward Status
Input database alias             = TEST
Number of nodes have returned status                         = 1
Node number                      = 0
Rollforward status               = not pending
Next log file to be read          =
Log files processed              = S0000003.LOG - S0000003.LOG
Last committed transaction       = 2008-12-11-15.00.29.000000 UTC
```

DB20000I  The ROLLFORWARD command completed successfully.

For multipartition databases, the **ROLLFORWARD DATABASE** command must be executed from the catalog partition.

After completing these steps, your database recovery is complete. At this point, you should connect to the database and verify that its contents are as expected. If the virtual device copy used is needed for further processing after the TimeFinder/Snap restore process is
complete, the corresponding virtual devices must be returned to a “Ready” state. This can be accomplished by executing a command that looks something like this:

```
symld -g [DeviceGroup] ready [VirtualDeviceID] -vdev
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
- **VirtualDeviceID** — Identifies a virtual device, by ID, that is to be returned to a “Ready” state.

Alternately, if after the restore process completes, the virtual device/SLV device relationship is no longer needed, it can be terminated by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

---

The following sections describe how to restore a recoverable database from a disk-based replication copy that was made using EMC TimeFinder technology while the database was online. The procedures outlined in these sections were developed using a single partition DB2 database that resided on a single Symmetrix array; the device group configuration used was similar to the one shown in Figure 51 on page 206.

**Note:** When restoring a recoverable database from an online database backup copy, just the SLV devices that contain database data should be restored; the SLV devices that contain the transaction logs should be left as they are so that the logs can be used for roll-forward recovery.
Restoring a recoverable database from an online database back up image that was created with TimeFinder/Clone

If TimeFinder/Clone was used to create a back up copy of a DB2 database, the TimeFinder/Clone restore process can be used to return the database to the state it was in at the time the back up copy was made. To restore a recoverable DB2 database from an online back up copy that was created using TimeFinder/Clone, complete the following steps, in the order shown:

1. Verify that the clone target devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Active” and “Copied” state) by executing the following command at the host (as the system administrator or root user):

   `symclone -g [DeviceGroup] query`

   where:
   - `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

   For example, to determine if the clone target devices that have been assigned to a device group named `symdb2dg` are in a valid state for recovery, you would execute a command that looks like this:

   `symclone -g symdb2dg query`

   And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

   ```
   Device Group (DG) Name: symdb2dg
   DG's Type             : REGULAR
   DG's Symmetrix ID     : 000190300709

   Source Device                   Target Device            State     Copy
   --------------------------------- ---------------------------- ------------ ----
   Logical   Sym  Tracks    Tracks   Logical   Sym  Tracks   CGDP SRC <=> TGT  (%)
   --------------------------------- ---------------------------- ------------ ----
   DEV001    0600         0        0 tgt719    0719        0 XX.. Copied       100
   DEV002    060A         0        0 tgt70f    070F        0 XX.. Copied       100
   DEV003    0614         0        0 tgt723    0723        0 XX.. Copied       100
   DEV004    061E         0        0 tgt72d    072D        0 XX.. Copied       100
   Total           -------- --------                --------
   Track(s)             0        0                       0
   MB(s)              0.0      0.0                     0.0
   ```
Legend:

(C): X = The background copy setting is active for this pair.
    . = The background copy setting is not active for this pair.
(G): X = The Target device is associated with this group.
    . = The Target device is not associated with this group.
(D): X = The Clone session is a differential copy session.
    . = The Clone session is not a differential copy session.
(P): X = The pre-copy operation has completed one cycle
    . = The pre-copy operation has not completed one cycle

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

   DEACTIVATE DATABASE [DBAlias]
   
   where:
   
   • DBAlias — Identifies the alias of the database to be deactivated (stopped).
   
   or
   
   FORCE APPLICATION ALL
   
   or
   
   db2stop FORCE

   For multipartition databases, the DEACTIVATE DATABASE command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

   **CAUTION**

   The FORCE_APPLICATION_ALL command will terminate all connections to all databases under the current DB2 Database Manager instance’s control; the db2stop FORCE command will terminate all connections to all databases under the current instance’s control and then stop the instance. (If the db2stop FORCE command is issued and applications do not respond within the time frame specified by the start_stop_time DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be
3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

For example, to unmount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```bash
for i in 1 2 3 4; do umount /db2fs$i; done
```

When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

```bash
df
```

The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.

For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```bash
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Clone restore process by executing the following command at the host (as the system administrator or root user):
symclone -g [DeviceGroup] restore -tgt -noprompt

where:

- **DeviceGroup** — Identifies the SYMCLI device group that
  TimeFinder/Clone control operations are to be performed on.

For example, to restore a set of SLV devices that have been
assigned to a device group named **db2datadg** from their
corresponding clone target devices, you would execute a
command that looks like this:

```
symclone -g db2datadg restore -tgt -noprompt
```

When this command is executed, all data stored on the SLV
devices in the device group named **db2datadg** will be overlaid
with data stored on the associated clone target devices.

**WARNING**

*Do not initiate a TimeFinder/Clone restore operation on volume(s)
that contain transaction log files; if such an operation is performed,
roll-forward recovery will no longer be possible - all transaction log
records produced since the database was copied will overwritten by
the restore operation.*

6. Verify that the TimeFinder/Clone restore process completed
successfully by executing one of the following commands at the
host (as the system administrator or root user):

```
symclone -g [DeviceGroup] verify -restored
```

or

```
symclone -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that
  TimeFinder/Clone control operations are to be performed on.

For example, to verify whether the restore process completed for
all devices that have been assigned to a device group named
**db2datadg**, you would execute a command that looks like this:

```
symclone -g db2datadg verify -restored
```

And if the restoration process is complete, when this command is
executed you should see a message that looks something like this:
All devices in the group 'db2datadg' are in 'Restored' state.

Once the TimeFinder/Clone restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the clone target devices to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. If a version of Solutions Enabler earlier than 6.0 is being used, terminate the clone target/SLV device relationship by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] terminate -tgt -noprompt
```

where:
- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

However, if Solutions Enabler 6.0 and Enginuity 5671 or later are being used, the clone target/SLV device relationship can be sustained by executing the following command at the host (as the system administrator or root user) instead:

```
symclone -g [DeviceGroup] split -tgt -noprompt
```

where:
- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

For example, to terminate the clone target/SLV device relationship between devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:

```
symclone -g db2datadg terminate -tgt -noprompt
```

And when this command is executed, you should see a message that looks like this:

'Terminate' operation execution is in progress for device group 'db2datadg'. Please wait...

'Terminate' operation successfully executed for device group 'db2datadg'.
On the other hand, to split a set of clone target devices that have been assigned to a device group named `db2datadg` from their corresponding SLV devices, you would execute a command that looks more like this:

```
symclone -g db2datadg split -tgt -noprompt
```

When this command is executed, you should see a message that looks something like this:

- ‘Clone Split’ operation execution is in progress for device group 'db2datadg'. Please wait...
- ‘Clone Split’ operation successfully initiated for device group 'db2datadg'.

8. Activate all SLV-related volume groups used, as well as their associated logical volumes. The actual command used to activate a volume group is operating system dependent.

For example, to activate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the root user):

```
importvg -y 'db2vg1' hdiskpower8
importvg -y 'db2vg2' hdiskpower9
importvg -y 'db2vg3' hdiskpower10
importvg -y 'db2vg4' hdiskpower11
for i in 1 2 3 4; do varyonvg db2vg$i; done
```

On the other hand, to activate the same four volume groups on a Linux server, you might execute a set of commands that look more like this (as the root user):

```
vgimport db2vg1 /dev/emcpowerb
vgimport db2vg2 /dev/emcpowerc
vgimport db2vg3 /dev/emcpowerd
vgimport db2vg4 /dev/emcpowere
for i in 1 2 3 4; do vgchange -a y db2vg$i; done
```

9. Once the appropriate volume groups have been activated, remount all file systems that are used by the database. The actual command used to mount a file system is operating system dependent.

For example, to mount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

```
mount /dev/emcpowerb/mnt/db2fs1
mount /dev/emcpowerc/mnt/db2fs2
mount /dev/emcpowerd/mnt/db2fs3
mount /dev/emcpowere/mnt/db2fs4
```
for i in 1 2 3 4; do mount /db2fs$i; done

When this command is executed, you can verify that the file systems have been mounted by executing a command that looks like this:

df

The file systems used by the database should now appear in the list of active file systems returned.

10. At this point, the database has been returned to the state it was in at the time the back up copy was made. However, recovery will not be complete until the database image is initialized and records stored in the database’s transaction log files have been reapplied. Initialize the database image and place it in “Roll-Forward Pending” state by executing the following command at the host (as the database instance user):

    db2inidb [DBName] AS MIRROR

where:

- **DBName** — Identifies by name, the database that is to be initialized and placed in “Roll-Forward Pending” state.

For multipartition databases, the `db2inidb` command must be executed on all partitions that belong to the database specified. This can be accomplished by executing the following command instead:

    db2_all || "db2 db2inidb [DBName] AS MIRROR"

Note: If the command `db2stop force` was used to terminate all connections to the database, the command `db2start` must be executed before the `db2inidb` command will execute successfully.

For example, to initialize a recoverable database named **TEST** and place it in “Roll-Forward Pending” state, you would execute a command that looks like this:

    db2inidb test AS MIRROR

And when this command is executed, you should see a message that looks something like this:

    DBT1000I The tool completed successfully.
11. Once the database image has been initialized and placed in the “Roll-Forward Pending” state, perform a roll-forward recovery operation by executing one of the following commands at the host (as the database instance user):

   ROLLFORWARD DATABASE [DBName] TO END OF LOGS AND STOP

   or

   ROLLFORWARD DATABASE [DBName] TO [Timestamp] AND STOP

   where:

   - **DBName** — Identifies, by name, the database that roll-forward recovery is to be performed on.
   - **Timestamp** — Identifies a specific point-in-time to which all committed transactions are to be rolled forward (including the transaction committed precisely at that date and time).

   For example, to perform a roll-forward recovery operation on a recoverable database named **TEST**, you could execute a command that looks like this:

   ROLLFORWARD DATABASE test TO END OF LOGS AND STOP

   And when this command is executed, you should see a message that looks something like this:

   Rollforward Status

   Input database alias             = TEST
   Number of nodes have returned status               = 1
   Node number                      = 0
   Rollforward status               = not pending
   Next log file to be read          =
   Log files processed              = S0000003.LOG - S0000003.LOG
   Last committed transaction       = 2008-12-11-15.00.29.000000 UTC

   DB20000I  The ROLLFORWARD command completed successfully.

   For multipartition databases, the **ROLLFORWARD DATABASE** command must be executed from the catalog partition.

   After completing these steps, your database recovery is complete. If you did not terminate the clone target/SLV device relationship, you will still have a point-in-time copy of the database available on the
clone target devices used should you need to restore the database again. At this point, you should connect to the database and verify that its contents are as expected.

When the copy of the database stored on the clone target device is no longer needed, it can be deleted by terminating the clone target device/SLV device relationship; this is done by executing the following command at the host (as the system administrator or root user):

```
symclone -g [DeviceGroup] terminate -tgt -noprompt
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Clone control operations are to be performed on.

---

**Restoring a recoverable database from an online database back up image that was created with TimeFinder/Snap**

As with TimeFinder/Clone, if TimeFinder/Snap was used to create a back up copy of a DB2 database, the TimeFinder/Snap restore process can be used to return the database to the state it was in at the time the back up copy was made. To restore a recoverable DB2 database from an online back up copy that was created using TimeFinder/Snap, complete the following steps, in the order shown:

1. Verify that the virtual devices that contain the point-in-time copy of the database to be restored are in a valid state for recovery (the “Associated” and “CopyOnWrite” state) by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] query
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been assigned to a device group named **symdb2dg** are in a valid state for recovery, you would execute a command that looks like this:

```
symsnap -g symdb2dg query
```
And if the devices are in a state that will allow the point-in-time copy of the database to be restored when this command is executed, you should see a message that looks something like this:

Device Group (DG) Name: symdb2dg
DG's Type: REGULAR
DG's Symmetrix ID: 000190300709

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Sym</td>
<td>Tracks</td>
<td>Logical</td>
</tr>
<tr>
<td>DEV001</td>
<td>0600</td>
<td>655500</td>
<td>vtgt5c0</td>
</tr>
<tr>
<td>DEV002</td>
<td>060A</td>
<td>655500</td>
<td>vtgt5ca</td>
</tr>
<tr>
<td>DEV003</td>
<td>0614</td>
<td>655500</td>
<td>vtgt5d4</td>
</tr>
<tr>
<td>DEV004</td>
<td>061E</td>
<td>655500</td>
<td>vtgt5de</td>
</tr>
</tbody>
</table>

Total Track(s): 2622000
Total MB(s): 163875

Legend:

(G): X = The Target device is associated with this group,
. = The Target device is not associated with this group.

2. Terminate all connections to the database to be restored by executing one of the following commands at the host (as the database instance user):

DEACTIVATE DATABASE [DBAlias]

where:

- **DBAlias** — Identifies the alias of the database to be deactivated (stopped).

or

FORCE APPLICATION ALL

or

db2stop FORCE
For multipartition databases, the **DEACTIVATE DATABASE** command will attempt to deactivate all partitions that belong to the database specified. However, this command may successfully deactivate some partitions and fail to deactivate others. Therefore, you must ensure that all partitions of a multipartition database have been deactivated before continuing with this procedure.

**WARNING**

The **FORCE APPLICATION ALL** command will terminate all connections to all databases under the current DB2 Database Manager instance's control; the **db2stop FORCE** command will terminate all connections to all databases under the current instance's control and then stop the instance. (If the **db2stop FORCE** command is issued and applications do not respond within the time frame specified by the **start_stop_time** DB2 Database Manager configuration parameter — which, by default is 10 minutes — DB2 will crash the database and it will need to undergo crash recovery before it will be usable.) Therefore, caution should be exercised when using these commands in environments where multiple databases fall under a single instance's control.

3. Unmount all file systems used by the database to ensure that nothing associated with the SLV devices that are to be restored remains in server memory. The actual command used to unmount a file system is operating system dependent.

   For example, to unmount four file systems named `db2fs1`, `db2fs2`, `db2fs3`, and `db2fs4` on an AIX or Linux server, you would execute a command that looks something like this (as the root user):

   ```bash
   for i in 1 2 3 4; do umount /db2fs$i; done
   ```

   When this command is executed, you can verify that the file systems have been unmounted by executing a command that looks like this:

   ```bash
   df
   ```

   The file systems should no longer appear in the list of active file systems returned.

4. Once the appropriate file systems have been unmounted, deactivate all SLV-related volume groups used. Again, the actual command used to deactivate a volume group is operating system dependent.
For example, to deactivate four volume groups named `db2vg1`, `db2vg2`, `db2vg3`, and `db2vg4` on an AIX server, you would execute a set of commands that look something like this (as the system administrator or root user):

```bash
for i in 1 2 3 4; do varyoffvg db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

On the other hand, to deactivate the same four volume groups on a Linux server, you would execute a set of commands that look more like this (as the root user):

```bash
for i in 1 2 3 4; do vgchange -a n db2vg$i; done
for i in 1 2 3 4; do vgexport db2vg$i; done
```

5. Initiate a TimeFinder/Snap restore process by executing the following command at the host (as the system administrator or root user):

```bash
symsnap -g [DeviceGroup] restore --noprompt
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to restore a set of SLV devices that have been assigned to a device group named `db2datadg` from their corresponding virtual devices, you would execute a command that looks like this:

```bash
symsnap -g db2datadg restore --noprompt
```

When this command is executed, all data stored on the SLV devices in the device group named `db2datadg` will be overlaid with data stored on the associated virtual devices.

**WARNING**

Do not initiate a TimeFinder/Snap restore operation on volume(s) that contain transaction log files; if such an operation is performed, roll-forward recovery will no longer be possible - all transaction log records produced since the database was copied will be overwritten by the restore operation.

6. Verify that the TimeFinder/Snap restore process completed successfully by executing one of the following commands at the host (as the system administrator or root user):
Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases

Restoring a DB2 LUW Database Using TimeFinder Technology

```
symsnap -g [DeviceGroup] verify -restored
```

or

```
symsnap -g [DeviceGroup] query
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to verify whether the restore process completed for all devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:

```
symsnap -g db2datadg verify -restored
```

And if the restoration process is complete, when this command is executed you should see a message that looks something like this:

```
All devices in the group 'db2datadg' are in 'Restored' state.
```

Once the TimeFinder/Snap restore process has been initiated, any additional recovery operations that may need to be performed against the restored database image can be performed immediately, while data is copied from the SAVDEV devices used to the appropriate SLV devices; any tracks accessed that have not yet been copied will be copied on demand.

7. When the TimeFinder/Snap restore process is initiated, both the VDEV devices and the corresponding SLV devices are set to a “Not-Ready” state (that is, they are offline to host activity). Once the restore operation commences, all source devices are placed in a “Ready” state. When the restore process is complete, terminate the existing internal copy session between the source and virtual/save devices by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -restored -noprompt
```

where:

- `DeviceGroup` — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to terminate the internal copy session that has been established for devices that have been assigned to a device group named `db2datadg`, you would execute a command that looks like this:
sym snap -g db2data dg terminate -restored -noprompt

And when this command is executed, you should see a message that looks something like this:

'Terminate' operation execution is in progress for device group 'db2data dg'. Please wait...

'Terminate' operation successfully executed for device group 'db2data dg'.

8. Verify that the virtual devices that contain the point-in-time copy of the database have been returned to a valid state for recovery (the “Associated” and “CopyOnWrite” states) by executing the following command at the host (as the system administrator or root user):

sym snap -g [DeviceGroup] query

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.

For example, to determine if the virtual devices that have been assigned to a device group named `db2data dg` have been returned to a valid state for recovery, you would execute a command that looks like this:

sym snap -g sym db2data dg query

And when this command is executed, you should see a message that looks something like this:

---

<table>
<thead>
<tr>
<th>Source Device</th>
<th>Target Device</th>
<th>State</th>
<th>Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Sym Tracks</td>
<td>Logical Sym Tracks</td>
<td>Changed Tracks SRC &lt;=&gt; TGT (%)</td>
<td></td>
</tr>
<tr>
<td>DEV001 0600 655500 vtgt5c0</td>
<td>DEV001 05C0</td>
<td>0 CopyOnWrite 0</td>
<td></td>
</tr>
<tr>
<td>DEV002 060A 655500 vtgt5ca</td>
<td>DEV002 05CA</td>
<td>0 CopyOnWrite 0</td>
<td></td>
</tr>
<tr>
<td>DEV003 0614 655500 vtgt5d4</td>
<td>DEV003 05D4</td>
<td>0 CopyOnWrite 0</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Track(s) 2622000</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
MB(s)  163875  0

Legend:

(G): X = The Target device is associated with this group,
    . = The Target device is not associated with this group.

9. Activate all SLV-related volume groups used, as well as their
    associated logical volumes. The actual command used to activate
    a volume group is operating system dependent.

    For example, to activate four volume groups named \texttt{db2vg1},
    \texttt{db2vg2}, \texttt{db2vg3}, and \texttt{db2vg4} on an AIX server, you would
    execute a set of commands that look something like this (as the
    root user):

    \begin{verbatim}
    importvg -y'\texttt{db2vg1}' hdiskpower8
    importvg -y'\texttt{db2vg2}' hdiskpower9
    importvg -y'\texttt{db2vg3}' hdiskpower10
    importvg -y'\texttt{db2vg4}' hdiskpower11
    for i in 1 2 3 4; do varyonvg db2vg$i; done
    \end{verbatim}

    On the other hand, to activate the same four volume groups on a
    Linux server, you might execute a set of commands that look
    more like this (as the root user):

    \begin{verbatim}
    vgimport db2vg1 /dev/emcpowerb
    vgimport db2vg2 /dev/emcpowerc
    vgimport db2vg3 /dev/emcpowerd
    vgimport db2vg4 /dev/emcpowere
    for i in 1 2 3 4; do vgchange -a y db2vg$i; done
    \end{verbatim}

10. Once the appropriate volume groups have been activated,
    remount all file systems that are used by the database. The actual
    command used to mount a file system is operating system
    dependent.

    For example, to mount four file systems named \texttt{db2fs1}, \texttt{db2fs2},
    \texttt{db2fs3}, and \texttt{db2fs4} on an AIX or Linux server, you would
    execute a command that looks something like this (as the root
    user):

    \begin{verbatim}
    for i in 1 2 3 4; do mount /db2fs$i; done
    \end{verbatim}

    When this command is executed, you can verify that the file
    systems have been mounted by executing a command that looks like this:

    \texttt{df}
The file systems used by the database should now appear in the list of active file systems returned.

11. At this point, the database has been returned to the state it was in at the time the back up copy was made. However, recovery will not be complete until the database image is initialized and records stored in the database’s transaction log files have been reapplied. Initialize the database image and place it in “Roll-Forward Pending” state by executing the following command at the host (as the database instance user):

\[
\text{db2inidb [DBName] AS MIRROR}
\]

where:

- **DBName** — Identifies by name, the database that is to be initialized and placed in “Roll-Forward Pending” state.

For multipartition databases, the `db2inidb` command must be executed on all partitions that belong to the database specified. This can be accomplished by executing the following command instead:

\[
\text{db2_all || “db2 db2inidb [DBName] AS MIRROR”}
\]

Note: If the command `db2stop force` was used to terminate all connections to the database, the command `db2start` must be executed before the `db2inidb` command will execute successfully.

For example, to initialize a recoverable database named TEST and place it in “Roll-Forward Pending” state, you would execute a command that looks like this:

\[
\text{db2inidb test AS MIRROR}
\]

And when this command is executed, you should see a message that looks something like this:

\[
\text{DBT1000I  The tool completed successfully.}
\]

12. Once the database image has been initialized and placed in the “Roll-Forward Pending” state, perform a roll-forward recovery operation by executing one of the following commands at the host (as the database instance user):

- \[
\text{ROLLFORWARD DATABASE [DBName] TO END OF LOGS AND STOP}
\]

or
ROLLFORWARD DATABASE [DBName] TO [Timestamp] AND STOP

where:

- **DBName** — Identifies by name, the database that roll-forward recovery is to be performed on.
- **Timestamp** — Identifies a specific point-in-time to which all committed transactions are to be rolled forward (including the transaction committed precisely at that date and time).

For example, to perform a roll-forward recovery operation on a recoverable database named **TEST**, you could execute a command that looks like this:

```
ROLLFORWARD DATABASE test TO END OF LOGS AND STOP
```

And when this command is executed, you should see a message that looks something like this:

```
Rollforward Status

Input database alias             = TEST
Number of nodes have returned status                         = 1

Node number                      = 0
Rollforward status               = not pending
Next log file to be read          =
Log files processed              = S0000003.LOG - S0000003.LOG
Last committed transaction       = 2008-12-11-15.00.29.000000 UTC

DB20000I  The ROLLFORWARD command completed successfully.
```

For multipartition databases, the **ROLLFORWARD DATABASE** command must be executed from the catalog partition.

After completing these steps, your database recovery is complete. At this point, you should connect to the database and verify that its contents are as expected. If the virtual device copy used is needed for further processing after the TimeFinder/Snap restore process is complete, the corresponding virtual devices must be returned to a “Ready” state. This can be accomplished by executing a command that looks something like this:

```
symld -g [DeviceGroup] ready [VirtualDeviceID] -vdev
```

where:

- **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
◆ **VirtualDeviceID** — Identifies a virtual device, by ID, that is to be placed in “Ready” state.

Alternately, if after the restore process completes, the virtual device/SLV device relationship is no longer needed, it can be terminated by executing the following command at the host (as the system administrator or root user):

```
symsnap -g [DeviceGroup] terminate -noprompt
```

where:

◆ **DeviceGroup** — Identifies the SYMCLI device group that TimeFinder/Snap control operations are to be performed on.
This appendix outlines the steps that were followed to create the environment that was used to test the procedures documented in this TechBook.

- Steps used to create the test environment ........................................ 308
Steps used to create the test environment

The following shows the steps that were used to create the test environment that was used to develop the procedures outlined in this TechBook:

1. A file named `create_meta_members.txt` was created using a text editor. This file, which contained commands to create the devices needed on the Symmetrix, looked like this:

   ```plaintext
   create dev count=40, size = 4370 cyl, emulation=fba, config=2-Way-BCV-Mir;
   ... 
   create dev count=32, size = 4370 cyl, emulation=fba, config=VDEV;
   ... 
   create dev count=40, size = 4370 cyl, emulation=fba, config=2-Way-Mir;
   ... 
   create dev count=40, size = 4370 cyl, emulation=fba, config=2-Way-Mir;
   ... 
   create dev count=8, size = 4370 cyl, emulation=fba, config=VDEV;
   ```

2. Using information stored in the `create_meta_members.txt` file, the devices needed were created by executing the following SYMCLI command:

   ```plaintext
   symconfigure -sid 709 -f create_meta_members.txt commit -noprompt
   ```

   When this command was executed, the following output was produced:

   A Configuration Change operation is in progress. Please wait...

   Establishing a configuration change session............Established.
   Processing symmetrix 000190300709
   Performing Access checks..............................Allowed.
   Checking Device Reservations.........................Allowed.
   Submitting configuration changes.......................Submitted.
   Validating configuration changes.......................Validated.

   New symdevs: 520:766
   Initiating PREPARE of configuration changes..........Prepared.
   Initiating COMMIT of configuration changes...........Queued.
   COMMIT requesting required resources.................Obtained.
3. Next, a file named **form_metas.txt** was created using a text editor. This file, which contained commands to create the metadevices needed on the Symmetrix, looked like this:

```plaintext
# form BCV device metas
form meta from dev 520 config = striped, count = 10;
form meta from dev 52a config = striped, count = 10;
form meta from dev 534 config = striped, count = 10;
form meta from dev 548 config = striped, count = 10;

# form VDEV device metas
form meta from dev 5c0 config = striped, count = 10;
form meta from dev 5ca config = striped, count = 10;
form meta from dev 5d4 config = striped, count = 10;
form meta from dev 5de config = striped, count = 10;

# form STD device metas
form meta from dev 600 config = striped, count = 10;
form meta from dev 60a config = striped, count = 10;
form meta from dev 614 config = striped, count = 10;
form meta from dev 61e config = striped, count = 10;

# form TGT device metas
form meta from dev 70f config = striped, count = 10;
form meta from dev 719 config = striped, count = 10;
form meta from dev 723 config = striped, count = 10;
form meta from dev 72d config = striped, count = 10;
```
4. Using information stored in the `form_metas.txt` file, the metadevices needed were created by executing the following SYMCLI command:

```
symconfigure -sid 709 -f form_metas.txt commit -noprompt
```

When this command was executed, the following output was produced:

```
A Configuration Change operation is in progress. Please wait...

   Establishing a configuration change session...........Established.
   Processing symmetrix 000190100709
   Performing Access checks............................Allowed.
   Checking Device Reservations.........................Allowed.
   Submitting configuration changes....................Submitted.
   Locking devices........................................Locked.
   Validating configuration changes.....................Validated.
   Initiating PREPARE of configuration changes.........Prepared.
   Initiating COMMIT of configuration changes..........Queued.
   COMMIT requesting required resources...............Obtained.
   Step 004 of 009 steps.................................Executing.
   Step 005 of 009 steps.................................Executing.
   Step 005 of 009 steps.................................Executing.
   Step 009 of 009 steps.................................Executing.
   Local: COMMIT.........................................Done.
   Terminating the configuration change session........Done.
```

The configuration change session has successfully completed.

5. Then, a file named `map_meta_heads.txt` was created using a text editor. This file, which contained commands to map the metadevice heads to the appropriate directors, looked like this:

```
map dev 600 to dir 16c:0, lun = 102;
map dev 600 to dir 1c:0, lun = 102;

map dev 60a to dir 16c:0, lun = 103;
map dev 60a to dir 1c:0, lun = 103;

map dev 614 to dir 16c:0, lun = 104;
map dev 614 to dir 1c:0, lun = 104;

map dev 61e to dir 16c:0, lun = 105;
map dev 61e to dir 1c:0, lun = 105;
```
6. Using information stored in the `map_meta_heads.txt` file, the metadevice heads were mapped to the appropriate directors by executing the following SYMCLI command:

```
symconfigure -sid 709 -f map_meta_heads.txt commit -noprompt
```

When this command was executed, the following output was produced:

```
A Configuration Change operation is in progress. Please wait...

Establishing a configuration change session........Established.
Processing symmetrix 000190300709
Performing Access checks...............................Allowed.
Checking Device Reservations............................Allowed.
Submitting configuration changes.........................Submitted.
Locking devices............................................Locked.
Validating configuration changes.........................Validated.
Initiating PREPARE of configuration changes.............Prepared.
Initiating COMMIT of configuration changes..............Queued.
COMMIT requesting required resources.................Obtained.
Step 003 of 012 steps.................................Executing.
Step 009 of 012 steps.................................Executing.
Local: COMMIT............................................Done.
Terminating the configuration change session.........Done.
```

The configuration change session has successfully completed.

7. Next, the metadevice heads were masked so that they could be seen by the HBAs at the AIX server by executing the following SYMCLI commands:

```
symmask -sid 709 -wwn 10000000c9577e72 -dir 16c -p 0
    add devs 600,60a,614,61e -dynamic_lun

symmask -sid 709 refresh -wwn 10000000c960b5b2 -dir 1c -p 0
    add devs 600,60a,614,61e -dynamic_lun
```

8. The device group used to hold the database transaction logs was created and the appropriate SLV devices, BCV devices, clone TGT devices, and VDEV devices were added to it by executing the following SYMCLI commands:

```
symdg create db2logdg
symld -g db2logdg add dev 61e -sid 709
symbcv -g db2logdg associate dev 548
symld -g db2logdg add dev 5de -vdev
symld -g db2logdg add dev 72d -tgt
```
9. The device group used to hold the database data was created and the appropriate SLV devices, BCV devices, clone TGT devices, and VDEV devices were added to it by executing the following SYMCLI commands:

   symdg create db2datadg
   symld -g db2datadg addall -range 600:614 -sid 709
   symbcv -g db2datadg associateall -range 520:534
   symld -g db2datadg addall -range 5c0:5d4 -vdev
   symld -g db2datadg addall -range 70f:723

10. The /var/symapi/config/options file was edited and the SYMAPI_ALLOW_DEV_IN_MULT_GRPS option was uncommented and assigned the value ENABLE.

11. The device group used to hold both the database data and the transaction logs was created and the devices in both the db2datadg and db2logdg device groups were added to it by executing the following SYMCLI commands:

   symdg create symdb2dg
   symld -g db2datadg copyall symdb2dg
   symbcv -g db2datadg copyall symdb2dg
   symld -g db2datadg copyall symdb2dg -vdev
   symld -g db2datadg copyall symdb2dg -tgt
   symld -g db2logdg copyall symdb2dg -rename
   symbcv -g db2logdg copyall symdb2dg -rename
   symld -g db2logdg copyall symdb2dg -rename -vdev
   symld -g db2logdg copyall symdb2dg -rename -tgt

12. Finally, the metadevices were made available to the AIX server, a partition and EXT3 file system were created on each one, and the file systems were mounted so they could be used for testing.
This glossary contains terms related to disk storage subsystems. Many of these terms are used in this book.

**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).

**alternate track** A track designated to contain data in place of a defective primary track. See also “primary track.”

**B**

**BCV device** A standard Symmetrix device with special attributes that allow it to independently support applications and processes. BCVs are active production images that are logically or physically separate from the production volumes with no reliance on the production host, thus providing protection from physical or logical corruption. Once the BCV task is complete, the volume can be resynchronized with the production volume, reassigned to another production volume, or maintained “as is” for another task. See also ”standard device.”
Glossary

**BVC mirror**
A standard device mirror (one of M2, M3, or M4) assigned to the BVC device upon establishing or re-establishing a BVC pair. See also “establish,” “re-establish,” and “BVC pair.”

**BVC pair**
Consists of a standard device and a BVC device attached together.

**Business Continuance (BC) Processes**
Processes that allow customers to access and manage instant copies of Symmetrix standard devices. See also “establish,” “re-establish,” and “split.”

**Business Continuance Volume (BCV)**
See “BVC device.”

**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.

**channel director**
The component in the Symmetrix subsystem that interfaces between the host channels and data storage. It transfers data between the channel and cache.

**CKD**
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.

**concurrent established BCV pair**
The relationship that establishes two BCV devices as concurrent mirrors of a single standard device that allows two synchronized copies of the standard data to be created simultaneously.

**controller ID**
Controller identification number of the director the disks are channeled to for EREP usage. There is only one controller ID for Symmetrix.

**D**

**DASD**
Direct access storage device, a device that provides nonvolatile storage of computer data and random access to that data. A DASD is most commonly known as a magnetic disk device.
<table>
<thead>
<tr>
<th><strong>Glossary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>data availability</strong></td>
</tr>
<tr>
<td><strong>define BCV pair</strong></td>
</tr>
<tr>
<td><strong>delayed fast write</strong></td>
</tr>
<tr>
<td><strong>destage</strong></td>
</tr>
<tr>
<td><strong>device</strong></td>
</tr>
<tr>
<td><strong>device address</strong></td>
</tr>
<tr>
<td><strong>device number</strong></td>
</tr>
<tr>
<td><strong>diagnostics</strong></td>
</tr>
<tr>
<td><strong>director</strong></td>
</tr>
<tr>
<td><strong>disk director</strong></td>
</tr>
<tr>
<td><strong>dual-initiator</strong></td>
</tr>
<tr>
<td><strong>dynamic sparing</strong></td>
</tr>
</tbody>
</table>
Glossary

E

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.

establish  A Business Continuance process that assigns a BCV device as the next available mirror of a standard device.

established  The BCV pair condition where the BCV device and standard device are synchronized and functioning as a Symmetrix mirror. A BCV pair is established by the BCV commands establish and re-establish.

F

fast write  In Symmetrix, 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.

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.

G

gatekeeper  A small logical volume on a Symmetrix storage subsystem used to pass commands from a host to the Symmetrix storage subsystem. Gatekeeper devices are configured on standard Symmetrix disks.

GB  Gigabyte, $10^9$ bytes.
H

head and disk assembly A field replaceable unit in the Symmetrix subsystem containing the disk and actuator.

home address 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. See also “CKD.”

hypervolume extension 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.

I

ID Identifier, a sequence of bits or characters that identifies a program, device, controller, or system.

IML Initial microcode program loading.

incremental Establish A time-saving operation similar to an Establish. The source (R1) device copies to the target (R2) device only the new data that was updated on the source R1 device while the SRDF pair was split. Any changed tracks on the target (R2) device are also refreshed from the corresponding tracks on the source (R1) device. The R2 device is write disabled to the target host.

index marker Indicates the physical beginning and end of a track.

index point The reference point on a disk surface that determines the start of a track.

instant Split A method of splitting a BCV that improves the performance of a typical split operation by performing a quick foreground BCV split, which reduces the time the application needs to be frozen and is shorter than using a regular Split.

I/O device An addressable input/output unit, such as a disk device.

K

K Kilobyte, 1024 bytes.
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>least recently used algorithm (LRU)</strong></td>
<td>The algorithm used to identify and make available the cache space by removing the least recently used data.</td>
</tr>
<tr>
<td><strong>logical volume</strong></td>
<td>A user-defined storage device.</td>
</tr>
<tr>
<td><strong>long miss</strong></td>
<td>Requested data is not in cache and is not in the process of being fetched.</td>
</tr>
<tr>
<td><strong>longitude redundancy code (LRC)</strong></td>
<td>Exclusive OR (XOR) of the accumulated bytes in the data record.</td>
</tr>
<tr>
<td><strong>MB</strong></td>
<td>Megabyte, (10^6) bytes.</td>
</tr>
<tr>
<td><strong>mirrored pair</strong></td>
<td>A logical volume with all data recorded twice, once on each of two different physical devices.</td>
</tr>
<tr>
<td><strong>mirroring</strong></td>
<td>The Symmetrix maintains two identical copies of a designated volume on separate disks. Each volume automatically updates during a write operation. If one disk device fails, Symmetrix automatically uses the other disk device.</td>
</tr>
<tr>
<td><strong>physical ID</strong></td>
<td>Physical identification number of the Symmetrix director for EREP usage. This value automatically increments by one for each director installed in Symmetrix. This number must be unique in the mainframe system. It should be an even number. This number is referred to as the SCU_ID.</td>
</tr>
<tr>
<td><strong>primary track</strong></td>
<td>The original track on which data is stored. See also &quot;alternate track.&quot;</td>
</tr>
<tr>
<td><strong>promotion</strong></td>
<td>The process of moving data from a track on the disk device to cache slot.</td>
</tr>
<tr>
<td><strong>protected BCV Establish</strong></td>
<td>The process of moving all mirrors of locally-mirrored BCV devices to join the mirrors of a standard device.</td>
</tr>
<tr>
<td><strong>read hit</strong></td>
<td>Data requested by the read operation is in cache.</td>
</tr>
</tbody>
</table>
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>read miss</td>
<td>Data requested by the read operation is not in cache.</td>
</tr>
<tr>
<td>record zero</td>
<td>The first record after the home address.</td>
</tr>
<tr>
<td>re-establish</td>
<td>A BC process that reassigns a BCV device as the next available mirror of the standard device with which it was previously paired. The BCV mirror is updated with the data that was written to the standard device during the period that the BCV pair was split. The data that was written to the BCV device during the split is overwritten by data from the standard device.</td>
</tr>
<tr>
<td>save devices</td>
<td>Device configured for use in TimeFinder/Snap operations and SRDF/A DSE. These devices not mapped to the host that provide polled physical storage space for storing pre-update images of the source device change tracks and new writes during a TimeFinder/Snap virtual copy session.</td>
</tr>
<tr>
<td>SAVDEVs</td>
<td>A collection of Save devices used for Snap operation. Also called Snap Save Pool or Snap Save Device Pool (formerly known as SAVDEVs Pool).</td>
</tr>
<tr>
<td>SCSI adapter</td>
<td>Card in the Symmetrix subsystem that provides the physical interface between the disk director and the disk devices.</td>
</tr>
<tr>
<td>scrubbing</td>
<td>The process of reading, checking the error correction bits, and writing corrected data back to the source.</td>
</tr>
<tr>
<td>short miss</td>
<td>Requested data is not in cache, but is in the process of being fetched.</td>
</tr>
<tr>
<td>SLV device</td>
<td>A Symmetrix device configured for normal Symmetrix operation under a desired protection method (such as RAID 1, RAID 5, RAID-S, SRDF). See also “standard device.”</td>
</tr>
<tr>
<td>snap device</td>
<td>See “virtual devices (VDEVs)” and “save devices (SAVDEVs).”</td>
</tr>
<tr>
<td>snap pool</td>
<td>A collection of Save devices used for Snap operation. Also called Snap Save Pool or Snap Save Device Pool (formerly known as SAVDEVs Pool).</td>
</tr>
<tr>
<td>source volume (R1)</td>
<td>A Symmetrix logical volume that is participating in SRDF operations. It resides in the local Symmetrix system. All CPUs attached to the Symmetrix may access a source volume for read/write operations.</td>
</tr>
</tbody>
</table>
Glossary

All writes to this primary source volume are mirrored (copied to a secondary target volume) in another Symmetrix system, which can be remote. A source volume is not available for local mirroring or dynamic sparing operations.

**split**  
A Business Continuance process that removes the BCV mirror from the existing BCV pair and assigns the BCV mirror back to its original device address. The BCV device then holds an instant copy of the data from the standard device.

**SSID**  
For 3990 storage control emulations, this value identifies the physical components of a logical DASD subsystem. 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.

**standard device**  
A Symmetrix device configured for normal Symmetrix operation under a desired protection method (such as RAID 1, RAID 5, RAID-S, SRDF).

**storage control unit**  
The component in the Symmetrix subsystem that connects Symmetrix 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.

**Symmetrix Logical Volume (SLV)**  
See “SLV device.”

**T**

**Target volume (R2)**  
A Symmetrix logical volume that is participating in SRDF operations. It resides in the remote Symmetrix system. This secondary target volume is paired with a primary source volume in the local Symmetrix system and receives all write data from its mirrored pair. This volume is not accessed by user applications during normal I/O operations. A target volume is not available for local mirroring or dynamic sparing operations.

**U**

**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

virtual devices (VDEVs)  Host-accessible devices containing track-level location information (pointers), which indicate where the copy session data is located in the physical storage. Virtual devices consume minimal physical disk storage, as they store only the address pointers to the data stored on the source device or a pool of Save devices. Using virtual devices, TimeFinder/Snap operations provide instant snapshot copies.

volume  A general term referring to a storage device. In the Symmetrix subsystem, a volume corresponds to a single disk device.

W

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.
A
Adaptive copy 101
ALTER DATABASE 61
ALTER TABLESPACE 59
Alter Tablespace dialog box 60
Asynchronous SRDF 101

B
BCV 119, 127, 193
device, definition of 313
mirrors 314
volumes, definition of 314, 320
BCV control operations
full establish 123
full restore 123
incremental establish 123
incremental restore 123
split 123
Buffer pools 43

C
Cache 90, 100
Change Tracker 88, 98
CKD 104
COMMIT 63
Composite groups 101
Con group trip 102
Concurrent SRDF 106
Configure
Databases 82
Instance 79
Servers 66, 73

Consistency group 104
Consistency groups 89, 101, 102, 103, 104, 110, 127
Containers 43
Crash recovery 200
CREATE DATABASE 44
Create Database Wizard 44
Create Table Space Wizard 58
CREATE TABLESPACE 58

D
Database 39
DB2 Administration Server (DAS) 40
DB2 family
DB2 Data Warehouse Edition (DWE) 26
DB2 Enterprise Server Edition (ESE) 24
DB2 Everyday 22
DB2 Express 23
DB2 Express-C 23
DB2 for i5/OS 26
DB2 for z/OS 27
DB2 Personal Edition 23
DB2 pureScale 25
DB2 Workgroup Server Edition (WSE) 24
DB2 objects
data objects 43
recovery objects 42
storage objects 42
system objects 42
DB2 Registry management tool 76
DB2 tools
Command Editor 33
Command Line Processor 36
Configuration Assistant 35
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>C</strong></td>
</tr>
<tr>
<td>Control Center 29</td>
</tr>
<tr>
<td>DB2_ENABLE_AUTOCONFIG_DEFAULT 53</td>
</tr>
<tr>
<td>DB2ADMINSERVER 40</td>
</tr>
<tr>
<td>DB2INSTANCE 39</td>
</tr>
<tr>
<td>db2set 74</td>
</tr>
<tr>
<td>db2unbind.lst 52</td>
</tr>
<tr>
<td>DBM Configuration dialog box 80</td>
</tr>
<tr>
<td>defining BCV pairs 315</td>
</tr>
<tr>
<td>Dependent-write consistency 104, 110</td>
</tr>
<tr>
<td>Device group 101</td>
</tr>
<tr>
<td>dftdbpath 47</td>
</tr>
<tr>
<td>DR 107</td>
</tr>
<tr>
<td><strong>E</strong></td>
</tr>
<tr>
<td>EMC TimeFinder</td>
</tr>
<tr>
<td>TimeFinder/Clone 117, 180</td>
</tr>
<tr>
<td>TimeFinder/Mirror 117, 180</td>
</tr>
<tr>
<td>TimeFinder/Snap 117, 180</td>
</tr>
<tr>
<td>Enginuity 88, 90, 94</td>
</tr>
<tr>
<td>Enginuity Consistency Assist 102, 103, 104</td>
</tr>
<tr>
<td>ESCON 90, 99</td>
</tr>
<tr>
<td><strong>F</strong></td>
</tr>
<tr>
<td>Failover, archival logging 71</td>
</tr>
<tr>
<td>FBA 104</td>
</tr>
<tr>
<td>Fibre Channel 90, 99</td>
</tr>
<tr>
<td>FICON 90</td>
</tr>
<tr>
<td><strong>G</strong></td>
</tr>
<tr>
<td>GET DATABASE CONFIGURATION 83</td>
</tr>
<tr>
<td>GET DATABASE MANAGER CONFIGURATION 79</td>
</tr>
<tr>
<td>Gigabit Ethernet 90, 99</td>
</tr>
<tr>
<td><strong>H</strong></td>
</tr>
<tr>
<td>HADR 24</td>
</tr>
<tr>
<td><strong>I</strong></td>
</tr>
<tr>
<td>I/O page cleaners 64</td>
</tr>
<tr>
<td>IBMDEFAULTTBP 50</td>
</tr>
<tr>
<td>Infinite logging 69</td>
</tr>
<tr>
<td>Instance 39</td>
</tr>
<tr>
<td>iSCSI 90</td>
</tr>
<tr>
<td><strong>L</strong></td>
</tr>
<tr>
<td>Log mirroring 70</td>
</tr>
<tr>
<td>logbufsiz 63</td>
</tr>
<tr>
<td><strong>M</strong></td>
</tr>
<tr>
<td>mirrors, BCV 314</td>
</tr>
<tr>
<td><strong>P</strong></td>
</tr>
<tr>
<td>Path failover 131</td>
</tr>
<tr>
<td>Path load balancing 130, 131</td>
</tr>
<tr>
<td>Path management 130</td>
</tr>
<tr>
<td>PowerPath 89, 102, 130</td>
</tr>
<tr>
<td><strong>R</strong></td>
</tr>
<tr>
<td>RA group 99, 102, 110</td>
</tr>
<tr>
<td>RAID 1 90</td>
</tr>
<tr>
<td>RAID 5 90</td>
</tr>
<tr>
<td>RAID 6 91</td>
</tr>
<tr>
<td>Registry variable 73</td>
</tr>
<tr>
<td>Remote adapter 99</td>
</tr>
<tr>
<td>Restartable databases 110</td>
</tr>
<tr>
<td>ROLLBACK 63</td>
</tr>
<tr>
<td>Rolling disaster 103, 104</td>
</tr>
<tr>
<td><strong>S</strong></td>
</tr>
<tr>
<td>Schema 51</td>
</tr>
<tr>
<td>Servers 39</td>
</tr>
<tr>
<td>Solutions Enabler 88, 89, 95</td>
</tr>
<tr>
<td>source volumes (R1) 319</td>
</tr>
<tr>
<td>split</td>
</tr>
<tr>
<td>definition of 320</td>
</tr>
<tr>
<td>SRDF 88, 98, 99, 100, 101, 104, 105, 107, 109, 110, 119</td>
</tr>
<tr>
<td>Establish and split operations 68</td>
</tr>
<tr>
<td>Suspend and resume operations 66</td>
</tr>
<tr>
<td>SRDF adaptive copy 101</td>
</tr>
<tr>
<td>SRDF Data Mobility 107</td>
</tr>
<tr>
<td>SRDF establish 108, 109</td>
</tr>
<tr>
<td>SRDF failback 111, 112</td>
</tr>
<tr>
<td>SRDF failover 111</td>
</tr>
<tr>
<td>SRDF restore 110</td>
</tr>
<tr>
<td>SRDF Split 109</td>
</tr>
<tr>
<td>SRDF/A 101</td>
</tr>
<tr>
<td>SRDF/AR 101</td>
</tr>
</tbody>
</table>

Using EMC TimeFinder to Back Up and Restore DB2 for Linux, UNIX, and Windows Databases
SRDF/CE 114
SRM 89
standard devices 320
SYMAP 88, 95
SYMCLI 95, 98, 103, 126, 128, 129
SYMCLI commands
   symclone 120
   symmir 122, 123
   symmsnap 194, 197
Synchronous SRDF 100
SYSCATSPACE 50
System catalog 50

T
Table space
   Automatic Storage 55
   Database Managed Space 55
   extent 54
   page 54
   System Managed Space 55
Tablespaces 43
target volumes (R2) 320
TEMPSPACE1 50

TimeFinder 89, 98, 107, 117
   consistent split 197
   restore operations 196
   split operations 195
TimeFinder/Clone 118, 126
TimeFinder/Mirror 117, 119
   reverse split 200
TimeFinder/Mirror control operations 122, 123
TimeFinder/Mirror establish 119
TimeFinder/Snap 118, 120
Transaction 63
Transaction logging 63

U
UPDATE DATABASE CONFIGURATION 83
UPDATE DATABASE MANAGER
   CONFIGURATION 79
USERSPACE1 50

V
VDEV 190
Virtual Provisioning 133