

# High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows

Learn HADR setup, administration, monitoring, and best practices

Use HACMP, MSCS, TSA with DB2 and DB2 HADR

Protect data with DB2 disaster recovery options



Whei-Jen Chen  
Masafumi Otsuki  
Paul Descovich  
Selvaprabhu Arumugharaj  
Toshihiko Kubo  
Yong Jun Bi





International Technical Support Organization

**High Availability and Disaster Recovery Options for  
DB2 on Linux, UNIX, and Windows**

February 2009

**Note:** Before using this information and the product it supports, read the information in “Notices” on page ix.

**Second Edition (February 2009)**

This edition applies to DB2 for Linux, UNIX, and Windows Version 9.5, Version 9.1.

**© Copyright International Business Machines Corporation 2009. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team that wrote this book .....	xi
Acknowledgements .....	xiii
Become a published author .....	xiv
Comments welcome .....	xiv
<b>Summary of changes</b> .....	xv
February 2009, Second Edition .....	xv
<b>Chapter 1. DB2 high availability and disaster recovery options</b> .....	1
1.1 Introduction .....	2
1.2 Hardware-based solutions .....	2
1.2.1 RAID technology .....	3
1.2.2 FlashCopy .....	7
1.2.3 Remote storage mirroring .....	8
1.3 Clustering solutions .....	10
1.3.1 Operating system dependent solutions .....	11
1.3.2 Operating system independent solutions .....	12
1.4 DB2 disaster recovery options .....	14
1.4.1 Overview .....	14
1.4.2 Backup and recovery .....	15
1.4.3 Log shipping solutions .....	19
1.4.4 DB2 HADR .....	20
1.4.5 Replication solutions .....	21
1.5 Hybrid use of shared disk cluster and HADR .....	25
1.5.1 Hybrid system .....	25
1.5.2 Case study of a hybrid system .....	27
<b>Chapter 2. HADR introduction</b> .....	31
2.1 High availability overview .....	32
2.1.1 Solutions for high availability .....	32
2.1.2 Types of high availability .....	33
2.2 HADR overview .....	33
2.3 DB2 logical log architecture .....	36
2.4 HADR architecture .....	39
2.5 Terminology .....	42

<b>Chapter 3. HADR setup</b> . . . . .	47
3.1 Requirements for setting up HADR . . . . .	48
3.2 Setup and configuration . . . . .	49
3.2.1 Preparing the environment . . . . .	49
3.2.2 Configuration using the Setup HADR wizard . . . . .	52
3.2.3 Command line setup . . . . .	71
3.3 Basic operation . . . . .	74
3.3.1 Starting up and shutting down . . . . .	74
3.3.2 Planned takeover . . . . .	79
3.3.3 Takeover by force . . . . .	83
3.4 Troubleshooting . . . . .	86
3.4.1 During setup . . . . .	87
3.4.2 After setup or during normal execution . . . . .	88
3.4.3 After an HADR disconnect or server failure . . . . .	90
3.4.4 Considerations while running HADR . . . . .	90
3.4.5 Re-establishing HADR after failure . . . . .	92
<b>Chapter 4. HADR administration and monitoring</b> . . . . .	99
4.1 Administering HADR systems . . . . .	100
4.2 Snapshot . . . . .	103
4.3 Monitoring HADR: db2pd . . . . .	106
4.4 Monitoring HADR: Administrative view and table function . . . . .	109
4.5 Monitoring HADR: db2diag . . . . .	113
4.6 Monitoring summary . . . . .	119
<b>Chapter 5. Automatic client reroute</b> . . . . .	121
5.1 Automatic client reroute overview . . . . .	122
5.1.1 DB2 automatic client reroute with HADR . . . . .	122
5.1.2 DB2 automatic client reroute in action . . . . .	123
5.2 Automatic client reroute tuning . . . . .	124
5.3 Automatic client reroute limitations . . . . .	128
5.4 Automatic client reroute configuration examples . . . . .	129
5.4.1 ACR with a non-HADR database . . . . .	130
5.4.2 ACR with an HADR database . . . . .	131
5.4.3 ACR with an HADR database in HACMP cluster . . . . .	132
5.5 Application programming to handle automatic client reroute . . . . .	134
5.5.1 Java application with JDBC driver . . . . .	134
5.5.2 Embedded SQL Program using C . . . . .	140
<b>Chapter 6. HADR best practices</b> . . . . .	143
6.1 DB2 HADR configuration parameters . . . . .	144
6.1.1 Basic configuration parameters . . . . .	144
6.1.2 Automatic client reroute configuration parameters . . . . .	149
6.2 DB2 HADR registry variables . . . . .	149

6.3 Recommendations and considerations . . . . .	151
6.3.1 DB2 transaction performance . . . . .	151
6.3.2 How to reduce the takeover time . . . . .	152
6.3.3 Seamless takeover . . . . .	153
6.3.4 Performance implications of HADR_TIMEOUT . . . . .	153
6.3.5 Applications with high logging rate . . . . .	153
6.3.6 Network considerations . . . . .	154
6.3.7 Avoiding transaction loss in an HADR with HA cluster software . . . . .	158
6.3.8 Avoiding transaction loss by using the peer window . . . . .	163
6.3.9 Index logging . . . . .	168
6.3.10 Read on the standby . . . . .	169
6.3.11 Backup from standby image with Flash Copy . . . . .	171
6.3.12 Replicating load data . . . . .	172
6.3.13 Log archive and HADR . . . . .	174
6.3.14 Database restore considerations . . . . .	174
<b>Chapter 7. DB2 and system upgrades . . . . .</b>	<b>175</b>
7.1 DB2 fix pack rolling upgrades . . . . .	177
7.1.1 Rolling upgrade on Microsoft Windows GUI . . . . .	177
7.1.2 Rolling upgrade on Linux Command Line . . . . .	191
7.2 DB2 migration (version upgrade) . . . . .	194
7.2.1 DB2 Migration on Microsoft Windows GUI . . . . .	196
7.2.2 DB2 migration on Linux command line . . . . .	224
7.3 Rolling OS/Application/DB2 configuration parameters . . . . .	231
<b>Chapter 8. DB2 with TSA . . . . .</b>	<b>233</b>
8.1 Overview . . . . .	234
8.1.1 TSA components . . . . .	234
8.1.2 Terminology of TSA . . . . .	236
8.2 How DB2 works with TSA . . . . .	238
8.3 Planning the high availability cluster . . . . .	241
8.4 Setting up TSA with DB2 9.1 . . . . .	243
8.4.1 Planning the cluster domain . . . . .	244
8.4.2 Configuration of TSA and DB2 . . . . .	246
8.5 Considerations for the db2nodes.cfg file . . . . .	272
8.6 DB2 9.5 High Availability Feature . . . . .	272
8.6.1 Cluster manager integration and interaction with DB2 . . . . .	273
8.6.2 Setting up TSA with DB2 9.5 . . . . .	274
<b>Chapter 9. DB2 and HACMP . . . . .</b>	<b>277</b>
9.1 Overview . . . . .	278
9.2 How DB2 works with HACMP . . . . .	279
9.3 Planning the HACMP cluster . . . . .	282
9.4 Setting up HACMP . . . . .	282

9.4.1	HACMP cluster setup planning	283
9.4.2	HACMP configuration	285
9.5	Considerations for db2nodes.cfg file	295
9.6	Tuning tips for quick failover	300
9.6.1	Failure detection time	301
9.6.2	Failover of the resources	301
<b>Chapter 10. DB2 with Microsoft Windows Server Cluster</b>		<b>307</b>
10.1	Server Cluster concepts	308
10.1.1	Types of clusters	308
10.1.2	Windows Server Cluster definitions	310
10.2	Minimum configuration and sample setup	311
10.3	Creating a Server Cluster	312
10.3.1	Adding the machines to the domain	313
10.3.2	Creating the cluster in the domain	315
10.4	Installing DB2 ESE	323
10.5	Adding resources to a cluster for DB2	324
10.6	Creating a DB2 instance	327
10.7	Manually configuring a DB2 instance in Windows Cluster	329
10.7.1	Adding the DB2 resource type	329
10.7.2	Creating a DB2 group	331
10.7.3	Creating a highly available IP address for the DB2 resource	334
10.7.4	Migrating the DB2 instance to the cluster environment	336
10.7.5	Adding a reference to the instance in the other nodes	337
10.7.6	Creating the highly available DB2 instance resource	338
10.8	Configuring a highly available DB2 instance using db2mscs	340
<b>Chapter 11. HADR with clustering software</b>		<b>345</b>
11.1	Overview: Why clustering software is needed	346
11.2	Automating HADR takeover with HACMP	350
11.2.1	HACMP and HADR planning	350
11.2.2	Step-by-step configuration overview	354
11.2.3	HADR setup	355
11.2.4	HACMP configuration	356
11.2.5	Prepare application server scripts	368
11.2.6	Joint test for HADR and HACMP	371
11.3	Automating HADR takeover with TSA on AIX	377
11.3.1	Architecture	378
11.3.2	Configuration	381
11.3.3	Administration of HADR with TSA	397
11.3.4	Failure test of HADR with TSA	401
11.4	Automating HADR takeover with TSA on Linux	405
11.4.1	Setting up HADR with TSA	405



11.4.2 Testing topology response to common failures. . . . .	442
--	-----

<b>Chapter 12. Configuring clusters using the DB2 9.5 High Availability Feature . . . . .</b>	<b>477</b>
12.1 db2haicu . . . . .	478
12.1.1 Prerequisites . . . . .	478
12.1.2 Usage . . . . .	479
12.1.3 Recommendations . . . . .	482
12.1.4 Troubleshooting . . . . .	482
12.2 DB2 HADR configuration for automatic failover with on AIX . . . . .	483
12.2.1 Architecture . . . . .	483
12.2.2 Configuration . . . . .	485
12.2.3 Administration . . . . .	495
12.2.4 Unplanned outages . . . . .	500
12.3 DB2 HADR configuration for automated failover on Linux . . . . .	508
12.3.1 Architecture . . . . .	509
12.3.2 Configuration . . . . .	510
12.3.3 Testing . . . . .	519
12.3.4 Administration . . . . .	533
12.4 DB2 with shared storage HA on AIX . . . . .	538
12.4.1 Architecture . . . . .	538
12.4.2 Configuration . . . . .	540
12.4.3 Administration . . . . .	551
12.4.4 Testing . . . . .	553
12.5 DB2 single partition HA with shared storage on Linux . . . . .	559
12.5.1 Architecture . . . . .	559
12.5.2 Configuration . . . . .	560
12.5.3 Testing . . . . .	577
12.6 DB2 DPF HA configuration with shared disk on AIX . . . . .	580
12.6.1 Architecture . . . . .	581
12.6.2 Preparation of the configuration . . . . .	583
12.6.3 Setting up the partitioned database . . . . .	585
12.6.4 Setting up the cluster domain by db2haicu . . . . .	605
12.6.5 Unplanned failure test . . . . .	610
12.6.6 Planned operations . . . . .	621
12.7 DB2 partitioned database HA configuration with shared disk on Linux .	626
12.7.1 Architecture . . . . .	627
12.7.2 Configuration . . . . .	628
12.7.3 Cluster domain configuration using db2haicu . . . . .	638
12.7.4 Unplanned operations . . . . .	653
12.7.5 Maintenance . . . . .	665
<b>Chapter 13. Q replication . . . . .</b>	<b>673</b>

13.1 Introduction . . . . .	674
13.2 Unidirectional setup . . . . .	675
<b>Chapter 14. Backup and Recovery . . . . .</b>	<b>713</b>
14.1 Single system view backup . . . . .	714
14.1.1 Using SSV . . . . .	714
14.1.2 Considerations . . . . .	720
14.2 Backup and restore database with snapshot backup . . . . .	720
14.2.1 DB2 ACS overview . . . . .	722
14.2.2 Storage layout considerations . . . . .	725
14.2.3 Prerequisites . . . . .	728
14.2.4 Installing DB2 Advanced Copy Services . . . . .	731
14.2.5 Activate DB2 ACS . . . . .	732
14.2.6 Configure DB2 ACS for IBM System Storage N Series with NFS . . . . .	734
14.2.7 Configure DB2 ACS for IBM System Storage SVC . . . . .	747
14.3 Recover database command . . . . .	759
14.4 Recovery object management . . . . .	791
<b>Appendix A. HACMP application server scripts . . . . .</b>	<b>809</b>
A.1 hadr_primary_takeover.ksh . . . . .	810
A.2 hadr_primary_stop.ksh . . . . .	812
A.3 hadr_monitor.ksh . . . . .	815
<b>Appendix B. DB2 and TSA configuration for ssh . . . . .</b>	<b>817</b>
B.1 Using DB2 with SSH . . . . .	818
B.2 Using TSA with SSH . . . . .	818
<b>Appendix C. TSA scripts . . . . .</b>	<b>825</b>
C.1 env file . . . . .	826
C.2 hadr_start.ksh . . . . .	827
C.3 hadr_stop.ksh . . . . .	831
C.4 hadr_monitor.ksh . . . . .	832
C.5 planned_takeover.ksh . . . . .	834
C.6 get_hadr_info.fnc . . . . .	836
<b>Related publications . . . . .</b>	<b>837</b>
IBM Redbooks publications . . . . .	837
Other publications . . . . .	837
Online resources . . . . .	840
How to get IBM Redbooks publications . . . . .	842
Help from IBM . . . . .	842
<b>Index . . . . .</b>	<b>843</b>

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:  
*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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.

This information contains 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.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX 5L™	General Parallel File System™	Redbooks (logo)  ®
AIX®	GPFS™	Server Advantage™
Balanced Warehouse™	HACMP™	Solid®
DB2 Connect™	IBM®	System i®
DB2 Universal Database™	Informix®	System p®
DB2®	MVS™	System Storage™
DS4000™	OpenPower®	System x™
DS6000™	OS/390®	System z®
DS8000™	POWER™	Tivoli®
ECKD™	pSeries®	TotalStorage®
Enterprise Storage Server®	pureXML™	WebSphere®
FlashCopy®	Redbooks®	z/OS®

The following terms are trademarks of other companies:

AMD, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

Snapshot, ONTAPI, FlexVol, NetApp, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

QLogic, and the QLogic logo are registered trademarks of QLogic Corporation. SANblade is a registered trademark in the United States.

LifeKeeper, SteelEye Technology, SteelEye, and the SteelEye logo are registered trademarks of SteelEye Technology, Inc. Other brand and product names used herein are for identification purposes only and may be trademarks of their respective companies.

J2EE, Java, JDBC, JVM, Solaris, Sun, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Visual Studio, Windows NT, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel Pentium, Intel Xeon, Intel, Pentium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

# Preface

As organizations strive to do more with less, IBM® DB2® for Linux®, UNIX®, and Windows® provides various built-in high availability features. DB2 further provides high availability solutions by leveraging enterprise system resources with broad support for clustering software such as HACMP™, TSA, and Microsoft® Windows Cluster Server.

This IBM Redbooks® publication describes DB2's high availability functions and features, focusing on High Availability Disaster Recovery (HADR) in the OLTP environment. The book provides a detailed discussion of HADR, including setup, configuration, administration, monitoring, and best practices.

We explain how to configure cluster software HACMP, TSA, and MSCS with DB2 and show how to use these products to automate HADR takeover.

DB2 also provides unprecedented enterprise-class disaster recovery capability. This book covers single system view backup, backup and restore with snapshot backup, as well as the db2recovery command, in detail.

## The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Rochester Center.

**Whei-Jen Chen** is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT Specialist.

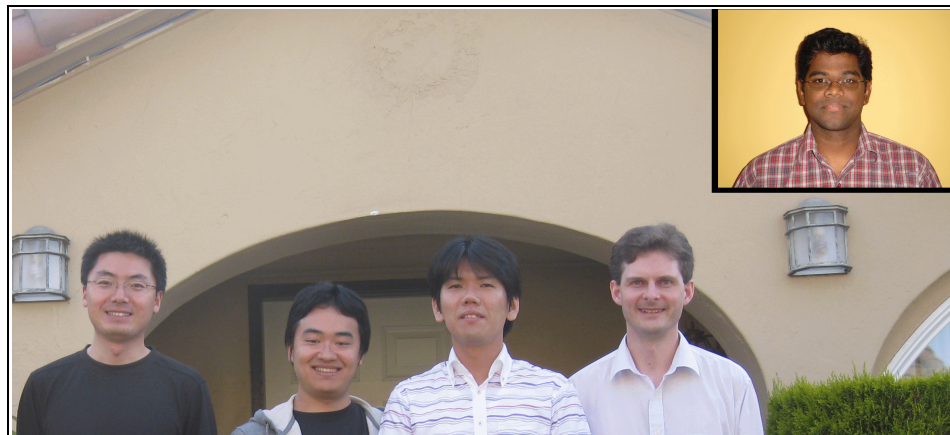
**Masafumi Otsuki** is an IT Specialist with IBM Systems Engineering Co., Ltd. in Japan. He has been working in DB2 technical support for two years focusing on DB2 high availability applications.

**Paul Descovich** works in a DB2 Systems Support role for IBM Global Services Australia, where most of his time is spent installing, configuring, and maintaining DB2 on midrange platforms. The rest of his time is spent interviewing architects to determine DB2 implementation standards, coordinating with UNIX sysadmins to get the necessary root authorities, and explaining how the DB2 System Support role is not the same as an Application DBA role. Paul started out application programming for DB2 v2r2 on MVS™ in 1991, then did DB2 System Support and DBA work from 1993, joining IBM in 1999.

**Selvaprabhu Arumugharaj** is an Advisory Software Engineer. He has worked for IBM for eight years supporting Information Management products including DB2 and Informix® database engines. He currently works on DB2 Down systems and Diagnostics (DSD) team supporting all DB2 customers, covering all engine components, and providing proactive support direction, diagnostic assistance, and solutions to one or more typically highly complex issues. He has solid knowledge of DB2 DPF and High Availability (HA) features.

**Toshihiko Kubo** is an IT Architect with DB2 for Linux, UNIX, and Windows. Currently, his key responsibility is in data service technical sales focusing on the areas of financial customers, high availability, Solid®, and pureXML™. He joined IBM in 1999. Toshihiko has experience in design of the mission-critical DBMS architectures and development of the clinical information system for hospitals.

**Yong Jun Bi** is a Senior IT Specialist currently working in Advanced Technical Support, IBM China. He focuses mainly on DB2 for LUW benchmark testing and High Availability and Disaster Recovery solutions. He also has rich experience in large scale database warehouse. Yong Jun is an IBM Certified Advanced Database Administrator.



*Left to right: Yong Jun, Masafumi, Toshihiko, Paul, Selvaprabhu*

## Acknowledgements

Thanks to the following people for their contributions to this project:

Steven Raspudic  
Dale McInnis  
Jimmy Stamkos  
Kelly Rodger  
Saurabh Sehgal  
Kenton Delathouwer  
IBM Toronto Laboratory

Yuke Zhuge  
Enrico Joedecke  
Dwaine Snow  
IBM Software Group

Yohichi Hara  
IBM Japan

Andreas Uhl  
Enrico Joedecke  
IBM Germany

Charles P. Brown  
IBM USA

Yvonne Lyon  
Sangam Racherla  
Deanna Polm  
Emma Jacobs  
Alex Osuna  
Mary Lovelace  
International Technical Support Organization, San Jose Center

Thanks also to the authors of the previous editions of this book:

- ▶ Authors of the first edition, *High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows*, published in October 4th, 2007, were:

Chandramouli Chandrasekaran  
Don Gneiting  
Gustavo Castro  
Paul Descovich  
Tomohiro Iwahashi

## Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You can have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts can help increase product acceptance and customer satisfaction. As a bonus, you can develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400



# Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes

for SG24-7363-01

for High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows

as created or updated on January 20, 2011.

## February 2009, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

### **New information**

- ▶ DB2 9.5 High Availability Feature
- ▶ DB2 9.5 HADR enhancements
- ▶ DB2 disaster recovery options

### **Changed information**

- ▶ Fix pack update





# DB2 high availability and disaster recovery options

In this chapter we provide an overview of different high availability and disaster recovery options in the market, where we briefly describe some technologies and products that can help you increase the availability of your DB2 applications.

We review these technologies and discuss, at a high level, how you can integrate DB2 with them. We also discuss some scenarios where these technologies might be suitable for your enterprise.

## 1.1 Introduction

DB2 provides several features that can increase high availability of the database servers, including these:

- ▶ High Availability and Disaster Recovery (HADR)
- ▶ Cluster manager integration with DB2 9.5 High Availability (HA) feature
- ▶ Automatic Client Reroute (ACR)
- ▶ SQL replication and WebSphere® Replication Server (also known as Q replication)
- ▶ DB2 9.5 Advanced Copy Service (ACS)
- ▶ Some basic DB2 features that also increase the availability of your operation, such as online backup and in-place reorganization

In addition to these options, DB2 supports a number of software and hardware offerings from IBM and other providers that you can use in conjunction with DB2 to strengthen high availability in your environment.

The decision on what products or features to use depends, as always, on the specific challenges of the environment, budget, complexity, time to implement, and degree of security required.

DB2 offers cost-effective alternatives to implement a very reliable solution. You can replace some of the DB2 high availability features with additional hardware and software, which, most of the time, drives costs up. For example, most high availability solutions implemented at the operating system level (clusters) heavily rely on shared disks, making storage the weak link in your operation. To overcome this limitation, you are required to implement storage mirrors that escalate costs.

## 1.2 Hardware-based solutions

Hardware solutions usually have the highest performance and security by copying or replicating whole disks. Using a hardware solution reduces the granularity and ability to determine when and what must be replicated. These solutions are also usually much more expensive than software solutions.

DB2 offerings are completely integrated with these hardware-based types of solutions. The redundant array of independent disks (RAID) technology, FlashCopy®, and remote storage mirroring are commonly used in a database system.

## 1.2.1 RAID technology

Redundant array of independent disks (RAID) technology consists of a group of individual disks that can be accessed as a whole to increase performance, fault tolerance, or both. While RAID technologies can be implemented at the software level, such implementations usually do not have the level of security or performance most database servers require. In this chapter, we discuss only hardware level RAID.

### RAID 0

RAID 0 is also known as *disk striping*. It refers to the usage of two or more disks, where the disk controller (known as a RAID controller) splits the I/O operations among the drives, creating a parallel write or read operation that increases performance but not reliability.

As Figure 1-1 shows, the RAID controller receives an 8 KB block to write. The controller splits the data into four 2 KB and writes the data parallel among all drives. RAID 0 provides no redundancy.

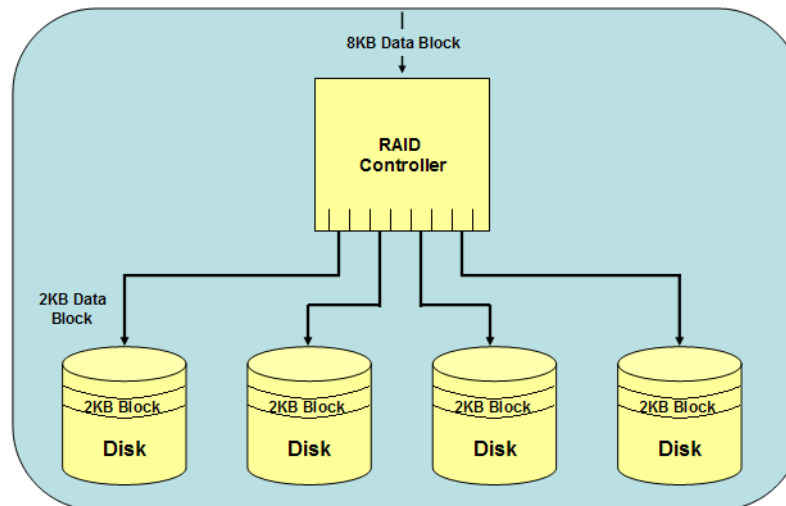


Figure 1-1 RAID 0 (disk striping) architecture

### RAID 1

RAID 1 is also known as *disk mirroring*. In RAID 1, the RAID controller replicates the writing operations from the main (or primary) disk to one or more spare disks called *mirrors*. Performance of write operations does not suffer, as the writing to the devices is accomplished in parallel.

Read operations are actually improved as the controller uses all devices to read from, reducing contention as illustrated in Figure 1-2.

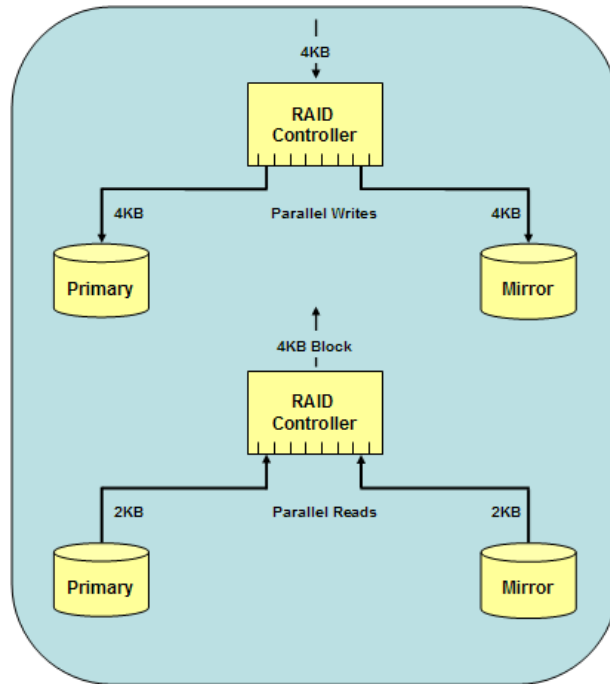


Figure 1-2 RAID 1 (disk mirroring) architecture

### RAID 3

Mirroring is great for security and very good for read performance, but it requires twice as much disk, which can be expensive. RAID 3 provides an alternative.

With RAID 3, a minimum of three disks is required. One of these disks is dedicated to store parity information about the data of the other two. In the event of a disk failure, the information in any of the disks can be re-created from the other two disks.

The downside of this technology is that the parity disk receives a heavy load, as any write operation requires a write on the parity disk. As illustrated in Figure 1-3, all writes require the computation of parity information, which is an additional CPU overhead. The more disks you have, the lower the percentage of disk space dedicated for parity.

Note that RAID 3 is not recommended for database storage, because of the single parity disk bottleneck.

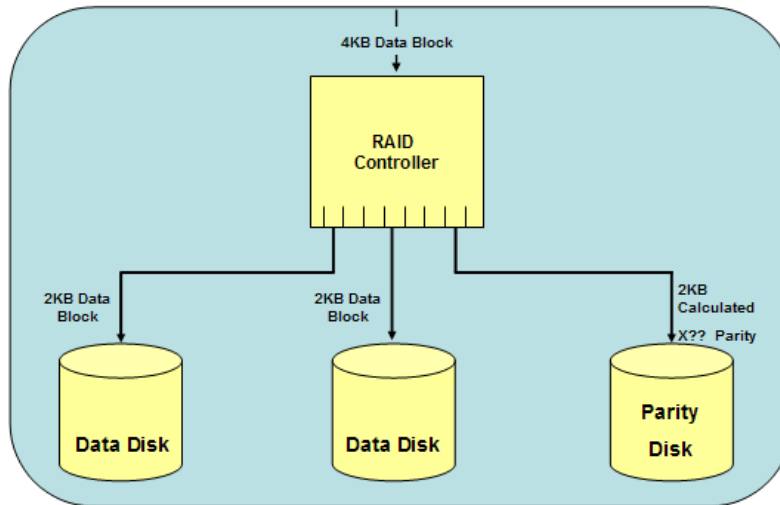


Figure 1-3 RAID 3 architecture in implementation

## RAID 5

RAID 5 is a variation of RAID 3. It has the same concept but instead of dedicating one single disk for parity, it spreads the parity among all the devices to balance the write load.

RAID 5 is a cost-effective alternative to mirror data. It provides redundancy at the expense of one physical disk per array and four I/O (2 reads and 2 writes) operations per I/O write request. As the number of physical disks in the array increases, the cost penalty for the array decreases, but the write penalty remains the same. Therefore, RAID 5 is not an optimal choice for online transaction processing (OLTP) database or random write I/O intensive systems.

Figure 1-4 shows how a basic RAID 5 system works.

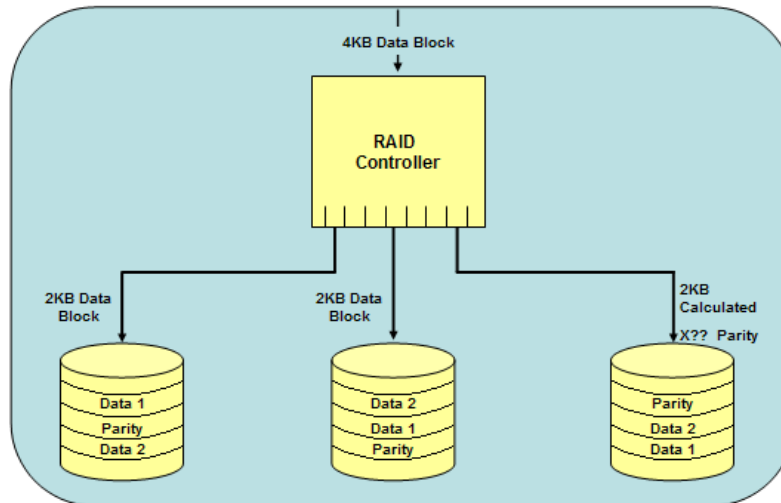


Figure 1-4 A basic RAID 5 example system

## RAID 10 and RAID 01

RAID 10 and RAID 01 (also known as 1+0 and 0+1) both use a combination of mirroring with striping. This is a winning combination for performance and fault tolerance but can be quite expensive.

The difference between RAID 10 and RAID 01 is that with RAID 10, individual disks are mirrored, and these mirrored groups are striped. On the other hand, with RAID 01, the mirror is applied to a whole striped volume. In other words, RAID 10 is a “stripe of mirrors”, and RAID 01 is a “mirror of stripes”. Because the failure of a single element in the stripe makes it unusable, the best alternative is to always stripe mirrored volumes, that is, RAID 10.

From a pure performance and fault tolerance point of view, RAID 10 (1+0) is the perfect match and the best option for databases. RAID 5 can be a very cost-effective alternative for non-OLTP database storage.

## When to use

In general, RAID can be considered for the following situations:

- ▶ When there are no concerns about site failures; this means, where the entire RAID would be lost or taken offline
- ▶ When fault tolerance, performance, or both are the most important factors



## DB2 integration

DB2 databases can benefit from using RAID. As mentioned before, RAID 5 is not suitable for databases running OLTP applications. However, using RAID 1 or RAID 10 for tablespaces increases the performance, tolerance to failure of your application, or both.

For more information about RAID, see the Web site:

[http://en.wikipedia.org/wiki/Redundant\\_array\\_of\\_independent\\_disks](http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks)

### 1.2.2 FlashCopy

FlashCopy is a feature of IBM System Storage™ subsystems, including Enterprise Storage Server® (ESS), Disk Subsystems (DS), and SAN Volume Controller (SVC). The FlashCopy function enables you to make point-in-time, full volume copies of data, with the copies immediately available for read or write access.

FlashCopy creates a copy of a source volume on the target volume. This copy is called a point-in-time copy. When you initiate a FlashCopy operation, a FlashCopy relationship is created between a source volume and a target volume. A FlashCopy relationship is a mapping of the FlashCopy source volume and a FlashCopy target volume. This mapping allows a point-in-time copy of that source volume to be copied to the associated target volume. The FlashCopy relationship exists between this volume pair from the time that you initiate a FlashCopy operation until the storage unit copies all data from the source volume to the target volume or you delete the FlashCopy relationship, if it is a persistent FlashCopy.

When the data is physically copied, a background process copies tracks from the source volume to the target volume. The amount of time that it takes to complete the background copy depends on the following criteria:

- ▶ The amount of data being copied
- ▶ The number of background copy processes that are occurring
- ▶ The other activities that are occurring on the storage subsystem

Other storage vendors also provide similar technologies, such as EMC TimeFinder/Snap, Hitachi ShadowImage, and NetApp® Snapshot™.

#### Advantages

Though it is expensive to implement FlashCopy solutions due to higher storage capacity requirements, FlashCopy has the following advantages:

- ▶ FlashCopy is very quick, and has a smaller impact to the production system.

- ▶ For mission critical applications, you can use FlashCopy to make several copies daily, and you can recover from logic errors quickly.
- ▶ FlashCopy target volumes can act as a point-in-time (PIT) backup source in a multi-level backup solution. For example, after you finish the FlashCopy, you can mount target volumes to a backup host and make a backup to a tape library from the backup host.

### **When to use**

If your business application is very critical and has high recovery time objective (RTO) requirements, you might want to consider FlashCopy. It can be used in a multi-level backup solution to decrease the impact to the production system and improve RTO.

### **DB2 integration**

DB2 9.5 ACS enables you to use the FlashCopy technology of a storage device to perform the data copying part of backup and restore operations. At the time of writing this book, the following hardware types are supported:

- ▶ IBM TotalStorage® SAN Volume Controller (SVC)
- ▶ IBM Enterprise Storage Server Model 800
- ▶ IBM System Storage DS6000™
- ▶ IBM System Storage DS8000™
- ▶ IBM System Storage N Series
- ▶ NetApp V-series

Chapter 14, “Backup and Recovery” on page 713 introduces how to use snapshot backup to backup and restore database.

## **1.2.3 Remote storage mirroring**

For years, mirroring and RAID have been the tools of choice to increase availability and performance at the hardware level. With ever increasing demands, many enterprises must not only be sure that their operations do not stop if a site failure occurs, but also require fast access to data at a remote location. Remote storage mirroring combines mirroring and replication in enterprise-level storage, allowing enterprises to provide solutions for these and other requirements.

The concept behind this technology is to have two or more disk arrays in boxes (also referred to as *Storage Enclosures*), located in geographically remote locations. These boxes are connected using *Dark fiber* or similar technologies and implement algorithms to replicate the data between the boxes.

Storage mirroring replicates all disk writes, including the log information. There are specific ways to update the remote disk change from brand to brand. In general, synchronous and asynchronous are two typical modes to update the remote disk.

Most of the time, remote storage mirroring has to be integrated with additional resources if automatic failover operations are required. Some examples of remote storage mirroring products are:

- ▶ IBM Peer to Peer Remote Copy (PPRC) and IBM Peer to Peer Remote Copy over eXtended Distances (PPRC-XD) on Enterprise Storage Server
- ▶ IBM Metro Mirror and IBM Global Mirror on IBM System Storage Disk Storage and IBM TotalStorage SAN Volume Controller
- ▶ Hitachi TrueCopy
- ▶ EMC Symmetrix Remote Data Facility (SRDF)

### **Advantages**

The advantages of remote storage mirroring are as follows:

- ▶ Several copies of data are kept in geographically remote locations.
- ▶ Copies are made at the hardware level, which usually improves performance and reliability.

### **When to use**

Use this technology when no data loss can be tolerated and the copy must be made with minimum delay.

### **DB2 integration**

Remote storage mirroring seamlessly integrates with DB2, allowing the enterprise to have a disaster recovery site providing continuous service. Additional technology, such as clustering, is required to automate the process of starting DB2 at the new site.

For more information, refer to the following Web sites:

- ▶ Dark fiber:  
[http://en.wikipedia.org/wiki/Dark\\_fiber](http://en.wikipedia.org/wiki/Dark_fiber)
- ▶ PPRC:  
[http://en.wikipedia.org/wiki/Peer\\_to\\_Peer\\_Remote\\_Copy](http://en.wikipedia.org/wiki/Peer_to_Peer_Remote_Copy)

## 1.3 Clustering solutions

Clustering is a technique that has been employed for a long time to increase the resilience, performance, and availability of the solutions implemented in servers. A *cluster* is a virtual machine made up of two or more physical machines (or nodes), each with various subcomponents registered to the cluster configuration as resources. The functionality of the resources and the nodes is logically bound together using additional software and hardware. Clusters create the concept of highly available resources. In the event of a node failure, resources on other nodes can take over the workload, minimizing the downtime.

The idea of clustering is to present to the users a single machine, when in fact the system has multiple nodes to serve the client applications. Many clusters act in an active/passive configuration where only one node performs work, with the others standing by as the backup in case of failure. Some cluster solutions are sophisticated enough to allow load balancing between the nodes in an active/active configuration, thus maximizing the performance of the applications, and providing more cost-effective use of the resources.

The DB2 HA Feature enables integration between IBM Data Server and cluster managing software.

When you stop a database manager instance in a clustered environment, you must make your cluster manager aware that the instance is stopped. If the cluster manager is not aware that the instance is stopped, the cluster manager might attempt an operation such as failover on the stopped instance. The DB2 HA Feature provides infrastructure for enabling the database manager to communicate with your cluster manager when instance configuration changes, such as stopping a database manager instance, require cluster changes.

The DB2 HA Feature is composed of the following elements:

- ▶ IBM Tivoli® System Automation for Multiplatforms (SA MP) Base Component is bundled with IBM Data Server on AIX® and Linux as part of the DB2 HA Feature, and integrated with the DB2 installer.
- ▶ In a clustered environment, some database manager instance configuration and administration operations require related cluster configuration changes. The DB2 HA Feature enables the database manager to automatically request cluster manager configuration changes whenever you perform certain database manager instance configuration and administration operations.
- ▶ DB2 High Availability Instance Configuration Utility (**db2haicu**) is a text based utility that you can use to configure and administer your highly available databases in a clustered environment.

- ▶ The DB2 cluster manager API defines a set of functions that enable the database manager to communicate configuration changes to the cluster manager.

Clustering solutions can be broadly categorized into two types:

- ▶ Operating system dependent
- ▶ Operating system independent

### 1.3.1 Operating system dependent solutions

Most operating systems provide tools or products to create clusters. The techniques and concepts are similar, and the intention is always to present to the world a single virtual server, although the services can be spread among the nodes or members of the cluster.

In the event of a service or resource failure, cluster processes try to restart the resource in the affected node, and if that is not possible, then resources on another node take their place.

Irrespective of cluster implementation, connection-oriented applications, such as database applications, require embedded logic to detect lost connections and to retry those operations that were in doubt or unfinished. The advantage of the cluster is that the clustering software often provides features for the applications to continue working effectively with no transactional data loss when the transfer of services between the nodes is automated. This reduces downtime and human intervention.

DB2 supports the following operating system dependent cluster managing software:

- ▶ IBM HACMP for AIX
- ▶ Microsoft Windows Server® 2003 Clustering Services for 32-bit/64-bit Intel/AMD™ platforms (formerly MSCS) and Windows Compute Cluster Server 2003 for 64-bit Intel/AMD platforms
- ▶ HP Multi-Computer/ServiceGuard for HP-UX
- ▶ Sun™ Cluster for Solaris™

#### **Advantage**

The advantage of the operating system dependent clustering software is that it is highly integrated with the operating system.

## When to use

If you have a homogeneous architecture, and you also use operating system dependent clustering software to manage other applications, then the operating system dependent clustering software is a good choice.

## DB2 integration

DB2 supports all the operating system dependent clustering software listed above. Chapter 9, “DB2 and HACMP” on page 277 explains how to integrate DB2 with HACMP on AIX. Chapter 10, “DB2 with Microsoft Windows Server Cluster” on page 307 describes procedures to implement DB2 cluster with Microsoft Windows Cluster Server.

For more information about operating system dependent software, see the following references:

- ▶ IBM HACMP:  
[http://en.wikipedia.org/wiki/High\\_Availability\\_Cluster\\_Multiprocessing](http://en.wikipedia.org/wiki/High_Availability_Cluster_Multiprocessing)
- ▶ Microsoft Windows Server 2003 Clustering Services:  
<http://www.microsoft.com/windowsserver2003/technologies/clustering/default.aspx>
- ▶ Microsoft Windows Compute Cluster Server 2003:  
<http://technet2.microsoft.com/windowsserver/en/technologies/featured/ccs/default.aspx>  
<http://www.microsoft.com/windowsserver2003/ccs/default.aspx>
- ▶ HP Serviceguard:  
<http://h71028.www7.hp.com/enterprise/cache/6469-0-0-0-121.html>
- ▶ Sun Cluster:  
<http://www.sun.com/software/cluster/>

## 1.3.2 Operating system independent solutions

These solutions have characteristics similar to the operating system dependent clustering software. This type of clustering software can work with various operating systems. They usually provide more sophisticated features to manage and control the clusters. Some allow frameworks that manage and interchange resources among groups of clusters.

Because these solutions are highly specialized, sometimes they offer more functionality than their operating system dependent counterparts. Some of these tools are:

- ▶ Tivoli System Automation for Multiplatforms (SA MP)
- ▶ Veritas Cluster Server

## Advantages

The operating system independent clustering software has the following advantages:

- ▶ These solutions are platform independent.
- ▶ They offer a single interface across different platforms.

## When to use

You can use the operating system independent clustering software:

- ▶ When you have a heterogeneous environment with different hardware providers.
- ▶ If platform change is a possibility.
- ▶ If your OS does not offer a cluster solution or has restrictions that can be avoided with an operating system independent clustering software.
- ▶ If you require to centrally manage a complex set of existing base-level clusters on differing platforms, which individually comprise only part of a larger overall application, then a multi-tiered cluster management solution such as Tivoli System Automation (TSA) End-to-End Automation can plug in to existing base-level clusters without any requirement to swap architecture or vendors.

## DB2 integration

There are DB2 agents or resource types to integrate these cluster solutions with DB2.

With DB2 9.5, Tivoli SA MP base components are bundled with DB2 on Linux and AIX, and we recommend that you use the integrated Tivoli SA MP base component on Linux and AIX to manage your cluster environment. Chapter 8, “DB2 with TSA” on page 233 explains how to implement a DB2 cluster with Tivoli SA MP on Linux and AIX. Chapter 12, “Configuring clusters using the DB2 9.5 High Availability Feature” on page 477 has details on using DB2 9.5 HA feature to configure cluster on Linux and AIX.

For more information on operating system independent software, refer to the following Web sites:

- ▶ Tivoli System Automation:  
<http://www-306.ibm.com/software/tivoli/products/sys-auto-multi/>
- ▶ Veritas Cluster Server:

## 1.4 DB2 disaster recovery options

In addition to the high availability options, DB2 also provides useful features for disaster recovery (DR) solutions. Note that these features are not solely for DR solutions. In this book, we focus our discussion on database disaster recovery only, mainly restoring database service after disasters.

### 1.4.1 Overview

In general, disaster recovery means that you recover your critical business operations after disasters, including natural disasters and human-induced disasters.

In 1992, the SHARE user group in the United States, in combination with IBM, defined a set of Disaster Recovery tier levels. This was to address the need to properly describe and quantify various methodologies of successful Disaster Recovery implementations for the mission-critical computer systems.

The *Seven Tiers of Disaster Recovery* solutions offer a simple methodology to define the current service level, the current risk, and the target service level and target environment. Because data is the key component in the whole DR solution, it is worth considering DB2 DR options in the seven tiers of DR framework. Next, we define the seven tiers of DR solution:

- ▶ Tier 0: No off-site data  
Businesses with a Tier 0 Disaster Recovery solution have no Disaster Recovery Plan. There is no saved information, no documentation, no backup hardware, and no contingency plan. The length of recovery time in this instance is unpredictable. In fact, it might not be possible to recover at all.
- ▶ Tier 1: Data backup with no hot site  
Businesses that use Tier 1 Disaster Recovery solutions back up their data at an off-site facility. Depending on how often backups are made, they are prepared to accept several days to weeks of data loss, but their backups are secure off-site. However, this Tier lacks the systems on which to restore data.
- ▶ Tier 2: Data backup with a hot site  
Businesses using Tier 2 Disaster Recovery solutions make regular backups on tape. This is combined with an off-site facility and infrastructure (known as a hot site) in which to restore systems from those tapes in the event of a



disaster. This tier solution still results in having to recreate several hours to days worth of data, but it is less unpredictable in recovery time.

- ▶ Tier 3: Electronic vaulting

Tier 3 solutions utilize components of Tier 2. Additionally, some mission-critical data is electronically vaulted. This electronically vaulted data is typically more current than that which is shipped through the Pickup Truck Access Method (PTAM). As a result there is less data recreation or loss after a disaster occurs.

- ▶ Tier 4: Point-in-time copies

Tier 4 solutions are used by businesses that require both greater data currency and faster recovery than users of lower tiers. Rather than relying largely on shipping tape, as is common in the lower tiers, Tier 4 solutions begin to incorporate more disk-based solutions. Several hours of data loss is still possible, but it is easier to make such point-in-time (PIT) copies with greater frequency than data that can be replicated through tape-based solutions.

- ▶ Tier 5: Transaction integrity

Tier 5 solutions are used by businesses with a requirement for consistency of data between production and recovery data centers. There is little to no data loss in such solutions; however, the presence of this functionality is entirely dependent on the application in use.

- ▶ Tier 6: Zero or little data loss

Tier 6 Disaster Recovery solutions maintain the highest levels of data currency. They are used by businesses with little or no tolerance for data loss, and who must restore data to applications rapidly. These solutions have no dependence on the applications to provide data consistency.

- ▶ Tier 7: Highly automated, business-integrated solution

Tier 7 solutions include all the major components being used for a Tier 6 solution with the additional integration of automation. This allows a Tier 7 solution to ensure consistency of data above that of which is granted by Tier 6 solutions. Additionally, recovery of the applications is automated, allowing for restoration of systems and applications much faster and more reliably than would be possible through manual disaster recovery procedures.

## 1.4.2 Backup and recovery

Backup and recovery is the fundamental technology for a DR solution.

## Backup

Backup is the process of backing up database objects to disks or tapes. You can transfer a backup image to an off-site facility in a Tier 1 DR solution.

DB2 provides flexible backup methods to meet various business requirements. You can back up a whole database or selected table spaces, either offline or online. Furthermore, you can back up a database incrementally, which means that you only have to back up changed data since your last backup, it is faster than a full backup. DB2 also provides automatic backup function; you define your backup policy and DB2 backup database automatically, which can save a lot of work for DBAs.

Because DB2 9.5, in a partitioned database environment, you can now do a Single System View (SSV) backup; you can back up a database on all database partitions or a subset of database partitions with one **backup** command. All backup images have same timestamp and include the necessary logs to roll forward to a consistent point of time. In 14.1, “Single system view backup” on page 714, we provide details about SSV backup.

DB2 Advanced Copy Service (ACS) provides integrated support for storage hardware FlashCopy function; you can use a single DB2 **backup** command to do a snapshot backup. It finishes immediately and has minimal impact on your production. In 14.2, “Backup and restore database with snapshot backup” on page 720, we introduce snapshot backup and restore in detail. Snapshot backup can be used with remote storage mirroring in a Tier 4 DR solution.

For more information about backing up a database, refer to the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006150.html>

## Recovery

Recovery is the rebuilding of a database or table spaces after a problem such as media or storage failure, power interruption, or application failure. If you have backed up your database, or individual table spaces, you can rebuild them if they become damaged or corrupted in some way. If you have a hot site to recover off-site backup images, this is a Tier 2 DR solution.

There are three types of recovery:

- ▶ Crash recovery protects a database from being left in an inconsistent or unusable state when transactions (also called units of work) are interrupted unexpectedly.
- ▶ Version recovery is the restoration of a previous version of the database, using an image that was created during a backup operation.

- ▶ Rollforward recovery can be used to reapply changes that were made by transactions that were committed after a backup was made.

The DB2 database manager starts crash recovery automatically to attempt to recover a database after a power interruption. You can use version recovery or rollforward recovery to recover a damaged database.

You can restore a whole database or selected table spaces. In some cases, if you need bring certain key table spaces online faster than others, you can use the database rebuild function. Rebuilding a database is the process of restoring a database or a subset of its table spaces using a set of restore operations. The functionality provided with database rebuild makes DB2 more robust and versatile, and provides you with a more complete recovery solution.

For more information about recovery, refer to the DB2 Information Center:

- ▶ Recover:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0011798.html>

- ▶ Restore:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006237.html>

- ▶ Rollforward:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006256.html>

## Managing recovery objects

As you regularly back up your database, you can accumulate some very large database backup images, as well as many database logs and load copy images. Storing recovery objects can consume great amounts of storage space. After subsequent backup operations are run, you can delete the older recovery objects because they are no longer needed to restore the database. However, removing the older recovery objects can be time consuming, and you might accidentally damage recovery objects that are still needed.

DB2 makes it easy by providing two ways to delete recovery objects that are no longer required to restore the database:

- ▶ You can invoke the PRUNE HISTORY command with the AND DELETE parameter, or call the db2Prune API with the DB2PRUNE\_OPTION\_DELETE flag.
- ▶ You can configure the database manager to automatically delete unneeded recovery objects. In 14.3, “Recover database command” on page 759, we provide details about automatic management of recovery objects.

For more information about managing recovery objects, refer to the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0051365.html>

You can also use third party software, such as IBM Tivoli Storage Manager, to back up and restore a database, and to manage recovery objects. DB2 provides integrated support for TSM. For more information about backing up and restoring a database with TSM, refer to DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0024705.html>

## **Developing a backup and recovery strategy**

DB2 provides flexible options to back up and restore your database, however, a backup and recovery strategy is critical to smooth operation of a DR solution.

Before developing your backup and recovery strategy, first answer the following questions:

- ▶ Is the database be recoverable?
- ▶ How much time can be spent recovering the database? This is to define the recovery time objective (RTO).
- ▶ How much data can you afford to lost after recovery? This is to define the recovery point objective (RPO).
- ▶ How much time can pass between backup operations?
- ▶ How much storage space can be allocated for backup copies and archived logs?
- ▶ How many backup copies and archived logs should I keep?
- ▶ Are table space level backups sufficient, or are full database backups necessary?
- ▶ Do I need a hot site for disaster recovery?
- ▶ Should I configure a standby system, either manually or through high availability disaster recovery (HADR)?

A database recovery strategy must ensure that all information is available when doing database recovery. It has to include at least the following characteristics:

- ▶ Have a regular schedule for taking database backups. We recommend that you do offline full database backups or online full database backups, including logs, periodically, such as one per month or one per week, depending on your business requirements. Between two full database backups, you can do online incremental backups periodically.

Make sure that you archive database log files correctly and that they are available when you do recovery. If you have many databases in your production system, we recommend using a third party backup software such as TSM. For more information, refer to *Backing Up DB2 Using IBM Tivoli Storage Management*, SG24-6247.

- ▶ Have a regular schedule for backing up command scripts, applications, user-defined functions (UDFs), stored procedure code in operating system libraries, and load copies.
- ▶ Include a hot site facility and staff if you are designing a DR solution with a hot site. You also require a proper method to transfer backup images to the hot site, either by physical transportation or electronic vaulting.
- ▶ Include processes or procedures on how to recover your database after a disaster occurs.

For more information, refer to DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0005945.html>

### 1.4.3 Log shipping solutions

Implementing a Tier 6 or Tier 7 DR solution requires zero or little data loss. If you have an active backup DB2 server on the hot site, it is possible to utilize a log shipping solution to implement these solutions.

The transaction log includes all data change operations on a database. You can reapply transactions in logs to a standby database to capture production database changes, just as we do when rolling forward a database through logs.

Here are the basic procedures for implementing a log shipping solution:

1. Take a full backup of the production database (Archive log mode).
2. On the standby database server in the hot site, restore the database and leave it in rolling forward pending status.
3. Keep transferring archive transaction logs from the production database to the standby database through the electronic method.
4. Reapply transaction logs in the standby database server through the **roll forward** command without the **stop** option.
5. After the production server is down, roll forward the standby database to the end of logs and complete. The standby database is now ready to use.

The recovery point objective (RPO) depends on the availability of the transaction logs — if all logs are available when a disaster occurs, there is no data loss at all.

The recovery time objective (RTO) depends on the log shipping delay and the rolling forward time on the standby database. With careful planning, you can achieve RTO less than an hour.

Since DB2 8.2, DB2 provides the HADR feature to do replication of any logged database activity to a local or remote location automatically. Refer to 1.4.4, “DB2 HADR”.

### **Advantages**

The log shipping solution can achieve zero or less data loss, and can recover database operation in the hot site quickly.

### **When to use**

We recommend that you use DB2 HADR after DB2 8.2. So you would only use the log shipping solution in the old version, which does not support DB2 HADR.

## **1.4.4 DB2 HADR**

DB2 High Availability and Disaster Recovery (HADR) is a feature in DB2 Enterprise Server allowing replication of any logged database activity to a local or remote location.

DB2 HADR works in a similar fashion to the log shipping solutions, but all the work is made inside DB2 at the software level. A DB2 HADR primary database uses internal processes to ship database log buffers to an HADR standby database. A process at the standby server then replays the log records directly to the standby database. The standby database can be converted to the primary database and accessed by applications and users in the event of a disaster, or if the primary server fails.

### **Advantages**

The advantages of using DB2 HADR include:

- ▶ Ultra™ fast failover capability
- ▶ Negligible impact on performance
- ▶ Easy to set up and monitor
- ▶ Rolling upgrades without service interruption between non-major versions or changes requiring server recycling, and reduced change windows for other upgrades
- ▶ Transparent failover and failback for applications
- ▶ Easy integration with high availability (HA) clustering software

- ▶ Dramatically improved disaster recovery compared to conventional methods

### When to use

You can use DB2 HADR in situations where there is:

- ▶ A requirement for a disaster recovery site with fast failover.
- ▶ No committed transaction data loss, where the replica (standby) database must be continually kept up-to-date with the primary. This means that you have the benefits of two-phase commit between separate systems, without the crippling performance overheads of synchronous update activity.
- ▶ No downtime for system change windows. HADR allows the rolling DB2 FixPak applications and certain configuration parameter changes to both databases, and operating system maintenance/recycling windows with no downtime for DB2 clients.

DB2 HADR is very suitable for a Tier 6 and Tier 7 DR solution. In Chapter 2 to Chapter 7, we describe in detail how to set up and administrate DB2 HADR.

For more information on DB2 HADR, refer to the following IBM manual: *Data Recovery and High Availability Guide and Reference*, SC23-5848-01.

## 1.4.5 Replication solutions

Replication solutions have been in use for quite some time. There are multiple replication technologies to suit different environments. In this section we focus on software-based replication.

DB2 provides two different solutions that you can use to replicate data from and to relational databases: SQL replication and WebSphere Replication Server (also known as Q replication). In SQL replication, committed source changes are staged in relational tables before being replicated to target systems. In Q replication, committed source changes are written in messages that are transported through WebSphere MQ message queues to target systems.

DB2 also provides a solution called *event publishing* for converting committed source changes into messages in an XML format and publishing those messages to applications such as message brokers.

Though SQL replication and Q replication is mainly used in data consolidation and distribution, we can also use them to build a hot standby server or implement peer-to-peer replication in a DR solution, especially in a heterogeneous environment.

## SQL replication

The source system has a capture program that scans the DB2 database log files looking for records belonging to replicated tables and then places this information in temporary storage known as *staging tables*.

The Apply program reads data from the staging tables and replicates data to corresponding target tables. The Apply program uses DB2 SQL operations to do replication.

Figure 1-5 shows an overview of the SQL replication architecture. Inside the source server are a set of DB2 relational tables called *Capture control tables*. Information about sources is written to these tables. The Capture program, which runs on the source server, uses this information to know what data it is supposed to capture. Information about targets goes in the Apply control tables, which are typically on your target server. The Apply program, which is also typically on your target server, uses this information to know which targets it is supposed to write data to.

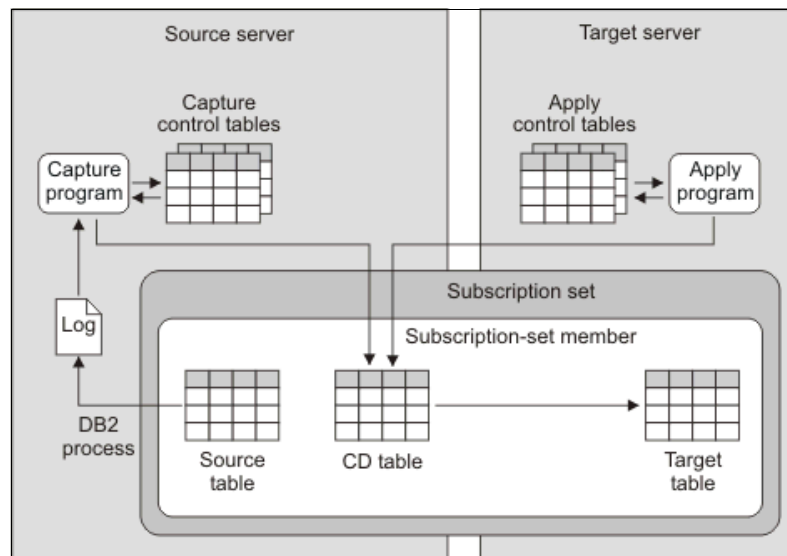


Figure 1-5 Infrastructure for a simple configuration in SQL replication

### Advantages

These are some advantages of DB2 SQL replication:

- ▶ It includes DB2 for OS/390® and z/OS® sources, targets, or both, in addition to DB2 for Linux, UNIX, and Windows sources, targets, or both.
- ▶ It can access multiple database management system (DBMS) types of sources and destinations.



- ▶ Replica databases are not disabled or hidden from client connectivity  
Client read-only access to replica database is recommended with SQL replication due to unidirectional replication.

### **When to use**

You can use SQL replication in the following situations:

- ▶ If you have a heterogeneous database provider environment, SQL replication can access more source and destinations than WebSphere Replication Server.
- ▶ When you have multiple targets from a single source, the information can be replicated or broadcast from the staging tables. This can suit geographic distribution for read-only or aggregate data for Management Information Systems or data warehousing.

For more information on SQL replication, refer to the following sources:

- ▶ *Information Integration: SQL Replication Guide and Reference*, SC19-1030-01
- ▶ SQL replication overview:  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.swg.im.iis.db.repl.sqlrepl.doc/topics/i1yrcncsqlreplovu.html>
- ▶ Replication solutions for common scenarios:  
[http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.websphere.i1.db2udb.replication.intro.doc/prod\\_overview/i1yrcintrsbdd.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.websphere.i1.db2udb.replication.intro.doc/prod_overview/i1yrcintrsbdd.html)

## **WebSphere Replication Server**

WebSphere Replication Server (also known as Q replication) is a replication solution that uses message queues in WebSphere MQ to transmit transactions between source and target databases.

The Q Capture program runs on the source server, and reads DB2 recovery logs for changes in the source data. These changes are then written to the WebSphere MQ queues. The Q apply program runs on the target server and receives the changes from the queues and writes to the targets.

WebSphere Replication Server is divided into three types:

- ▶ Unidirectional:
  - One-way replication from a source to a target
  - Possibility to configure multiple sources to a target or multiple targets to a source

- ▶ Bidirectional:  
Two-way replication between a source and a target table on two servers
- ▶ Peer to peer:  
Two-way replication between a source and a target table on two or more servers

Figure 1-6 describes a unidirectional architecture using WebSphere Replication Server. Note that reference is only made to a DB2 process on the source server; however, this does not preclude client connectivity on the target server. DB2 processes exist here with an active target database in order to accept the SQL transactions being converted from message queue format by the Q Apply program.

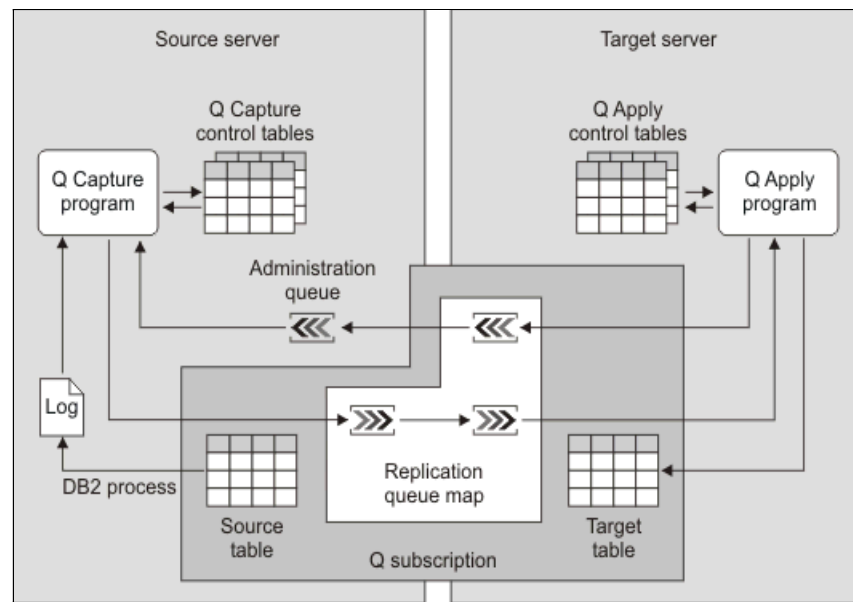


Figure 1-6 A simple configuration in WebSphere Replication Server

### **Advantages**

The advantages of WebSphere Replication Server are as follows:

- ▶ DB2 for OS/390 and z/OS sources, targets, or both; in addition to DB2 for Linux, UNIX, and Windows sources, targets, or both
- ▶ Supports high volumes of changes
- ▶ Low latency requirements
- ▶ Supports a large number of source/target tables

- ▶ Replicas/targets with conflict detection/resolution that can be updated
- ▶ Captured transactions can feed other applications

### ***When to use***

You can use WebSphere Replication Server:

- ▶ For capacity relief such as read/write database for both source and target, splitting the transaction load between the servers with an external load balance mechanism.
- ▶ Geographically distributed applications to update back to a central database or in the opposite direction with a central hub broadcasting to multiple distributed targets. Note that we recommend SQL replication for the latter configuration.
- ▶ For rolling upgrades.
- ▶ When a subset of columns and rows are required for replication.

For more information, refer to “Replication Scenarios for Common Solutions” on the Web at:

[http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod\\_overview/iityrcintrsbdd.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iityrcintrsbdd.html)

## **1.5 Hybrid use of shared disk cluster and HADR**

For a local HA solution HADR provides extremely fast failover because the standby database server can be in the same room. If you are building a Disaster Recovery (DR) solution, the standby servers can be located remotely. If you want to build a combined HA and DR solution, you normally have to combine HADR with another of the options, such as a shared disk solution.

This section describes the combination (hybrid) of these two solutions and how this hybrid system strengthens high availability of the database system. With HADR configured on the remote hot site, this configuration can be used to implement a Tier 6 or Tier 7 DR solution.

### **1.5.1 Hybrid system**

There are several considerations for a shared disk cluster. Because resources are shared between the cluster nodes, a shared disk configuration cannot provide services in the following situations:

- ▶ Crash or planned maintenance of the shared storage device
- ▶ Maintenance that requires stopping the cluster software
- ▶ FixPak update or configuration change that requires stopping the database instance
- ▶ Site disaster

By adding an HADR standby database to the shared disk cluster, continuous service is possible even in the cases mentioned previously. Here we call this system a *hybrid system*. Figure 1-7 illustrates a hybrid high availability system with shared disk cluster and HADR.

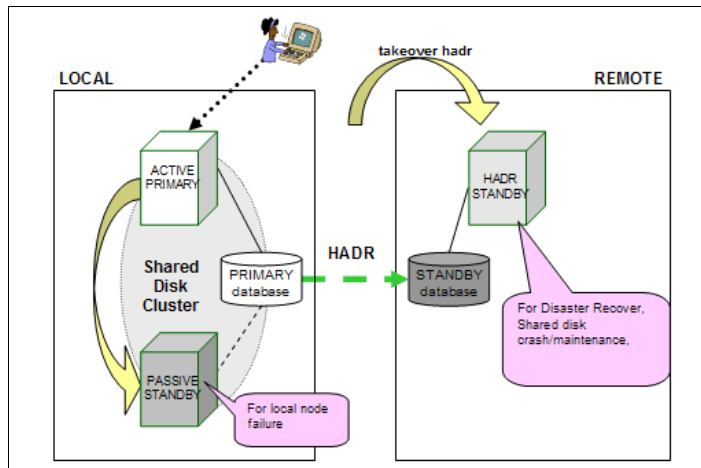


Figure 1-7 Hybrid high availability system with shared disk cluster and HADR

In this system configuration, there are three nodes with different roles:

- ▶ Active primary node:
 

This node provides service with shared disks. The DB2 instance is running on it. Also, it plays a role of HADR primary database, transferring log records to standby database.
- ▶ Passive (local) standby node:
 

If the active primary node fails, the local standby node takes over the resources, including shared disk.
- ▶ HADR (remote) standby node:
 

This is the node where the HADR standby database is running. When both the active primary node and the passive standby node are not available, this node can provide service. The HADR standby can be placed in the same local site as the other nodes, but it must be in a remote site for the purpose of disaster recovery.

## 1.5.2 Case study of a hybrid system

The following operations are required of a hybrid system when possible failure or maintenance in the database system occurs:

- ▶ Failure of the primary node:

The resource group, including the shared disk, is automatically failed over to the passive standby node, and the service is restarted. After failover, the HADR connection is re-established with the new active primary (passive standby), as shown in Figure 1-8.

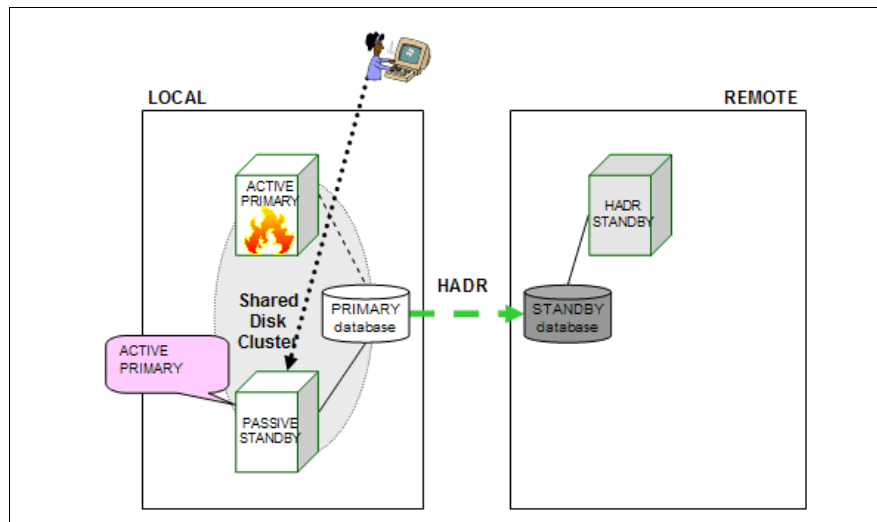


Figure 1-8 Failure of the primary node

- ▶ Crash or planned maintenance of the shared storage device:

If the shared disk is unavailable for any reason, the shared disk cluster cannot continue its services. In this case, the service can be taken over to the HADR standby — even when the shared disk has to be stopped for the purpose of maintenance, such as upgrading the microcode of storage devices, or for restructuring the storage configuration. You do not have to stop database services for applications, as shown in Figure 1-9.

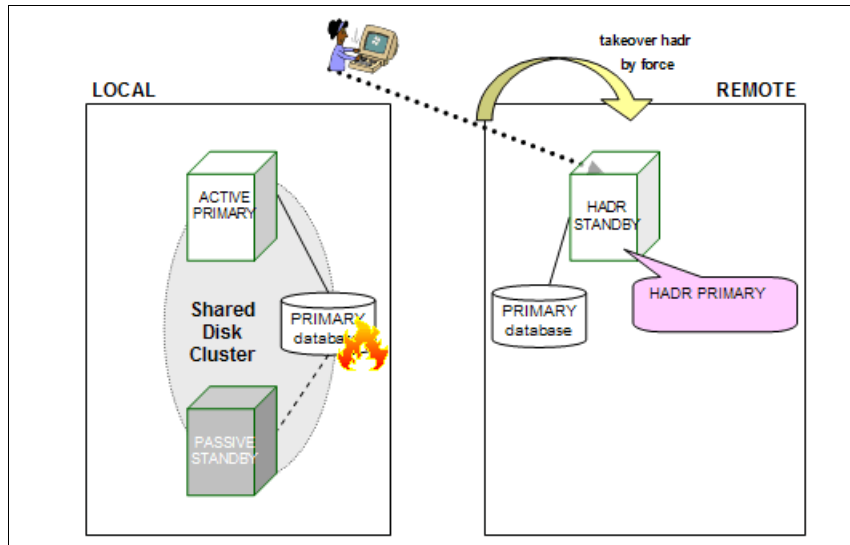


Figure 1-9 Crash or planned maintenance of shared storage device

► Cluster software maintenance:

When you are required to stop the cluster software to make some configuration changes and a failover test is required, you can fail over the services to the HADR standby database. Meanwhile, you can check the behaviors of the cluster software without any impact on the application services.

► Upgrading DB2 FixPak (Rotating upgrade):

You can upgrade DB2 FixPak with only an extremely short service downtime.

Complete the following steps to upgrade DB2 FixPak:

- a. Update FixPak on HADR standby and re-establish the HADR connection.
- b. Fail over to HADR standby. The downtime for the application is just a few seconds of taking over the HADR role.
- c. Update DB2 FixPak in the shared disk cluster (active primary node and passive standby node).
- d. Switch the HADR role back to the local cluster.

► Using standby database as staging environment:

It is frequently necessary to test new applications or new database functions on a database, which has an amount of data (or data itself) similar to the product environment. After the standby database is detached from the HADR pair, it is available for a staging environment with complete data replicated from the product environment. With the hybrid system, you still have redundancy for local failure while you are utilizing HADR standby as the staging environment.

The staging environment is a test environment/infrastructure that mirrors the production and allows you to test new or changed application or settings without affecting the production system and the user, as shown in Figure 1-10.

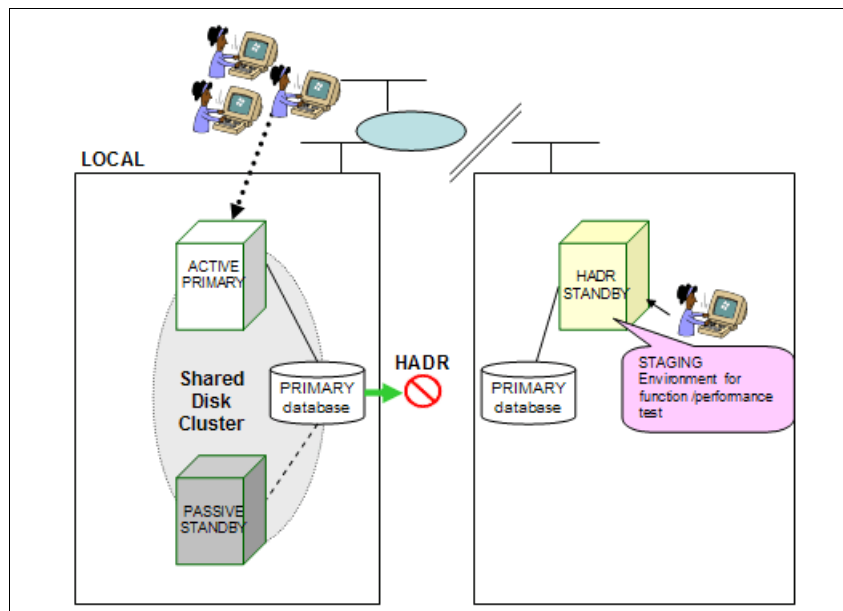


Figure 1-10 Use standby database as staging environment







## HADR introduction

Organizations today face a tough challenge in choosing an appropriate high availability solution that meets their business requirements and IT budgets.

Until recently, such solutions were based on proprietary systems, and involved significant investment in capital, time, and resources to assemble, integrate, test, manage, and support. This scenario has been changed dramatically with the introduction of software-based high availability solutions.

In this chapter we provide a general overview of an IBM software-based high availability solution: *DB2 High Availability Disaster Recovery (HADR)*.

We cover the following topics:

- ▶ High availability overview
- ▶ HADR overview
- ▶ Why HADR?
- ▶ HADR architecture
- ▶ Terminology

## 2.1 High availability overview

High availability (HA) is the term used to describe systems that run and are available to customers more or less all the time. For this to occur:

- ▶ Transactions must be processed efficiently, without appreciable performance degradations (or even loss of availability) during peak operating periods.
- ▶ Systems must be able to recover quickly when hardware or software failures occur, or when disaster strikes. DB2 has an advanced continuous checkpointing system and a parallel recovery capability to allow for extremely fast crash recovery.
- ▶ Software that powers the enterprise databases must be continuously running and available for transaction processing. To keep the database manager running, you must ensure that another database manager can take over if it fails. The concept of moving functionality from one server to another is variously known as *failover* or *takeover*, largely depending on the software vendor's terminology. Failover capability allows for the automatic transfer of workload from one system to another when there is a hardware failure.

### 2.1.1 Solutions for high availability

Now more than ever, customers are demanding a 24x7 operating environment. To implement this requirement, organizations must give high availability and disaster recovery a great deal of consideration.

Many customers use a hardware high availability (HA) approach, where all the application updates on one machine are synchronized to a backup machine. However, there are many advantages of using a software-based approach, chief among them being lower cost. IBM provides software-based HA solutions such as DB2 HADR, WebSphere Replication Server Q Replication, SQL Replication, and HA clustering software — HACMP and Tivoli System Automation (TSA).

HADR is one among several replication solutions offered in the DB2 product family. WebSphere Information Integrator and the DB2 database system include SQL replication and Q replication solutions that can also be used, in some configurations, to provide high availability. These functions maintain logically consistent copies of database tables at multiple locations. In addition, they provide flexibility and complex functionality such as support for column and row filtering, data transformation, updates to any copy of a table, and can also be used in partitioned database environments.

## 2.1.2 Types of high availability

There are two types of high availability:

▶ Continuous availability:

Continuous availability requires that the database engine be available for processing SQL transactions without any downtime. Such availability is usually implemented in mission-critical business applications. For this to happen, total redundancy is required, which means that you must have two systems that are fully independent of each other, both in terms of hardware and software.

▶ Failover availability:

Failover availability is differentiated from continuous availability by the fact that for some period of time, however small, the database engine or other critical server function is not available for transaction processing. The essential components for this type of solution are:

- Primary and standby systems
- Failure detection
- Data source movement

In the context of providing continuity for a database management system, the two systems (primary and standby) must each have access to copies of the database data, which clients can normally access from the primary. When the failure detection mechanism is triggered by a critical resource breaking on the primary, a failover takes place. In the failover process, the data source is moved from the primary to the standby system, and clients start accessing the data through that system. Note also that this process is seamless from the clients' perspective. Some solutions allow for the data source to be replicated, such that the primary data and the standby data are stored separately. Others have the data in a single location, which is shared between a primary and a standby system, but it is only accessed by one system at any given time.

This book focuses on solutions providing failover availability, although to some extent, certain scalability solutions also provide a type of continuous availability.

## 2.2 HADR overview

DB2 High Availability Disaster Recovery (HADR) is a database replication feature that provides a high availability solution for both partial and complete site failures. A database server can fail from any number of factors: environmental (power/temperature), hardware, network connectivity, software, or human intervention.

Recovery from the loss of a database server in a standard installation can require a machine reboot and database crash recovery, which interrupt the database services. By using DB2 HADR, you can safely minimize the downtime to only a few seconds. HADR protects against data loss by continually replicating data changes from a source database, called the primary, to a target database, called the standby. Furthermore, you can seamlessly redirect clients that were using the original primary database to the standby database (which becomes the new primary database) by using Automatic Client Reroute (ACR) and retry logic in the application. This seamless redirection is also possible using software solutions discussed in other chapters, which can manage IP address redirection.

Figure 2-1 illustrates the concept of HADR.

HADR transmits the log records from the primary database server to the standby server. The HADR standby replays all the log records to its copy of the database, keeping it synchronized with the primary database server. The standby server is in a continuous rollforward mode and is always in a state of near-readiness, so the takeover to the standby is extremely fast. Applications can only access the primary database and have no access to the standby database.

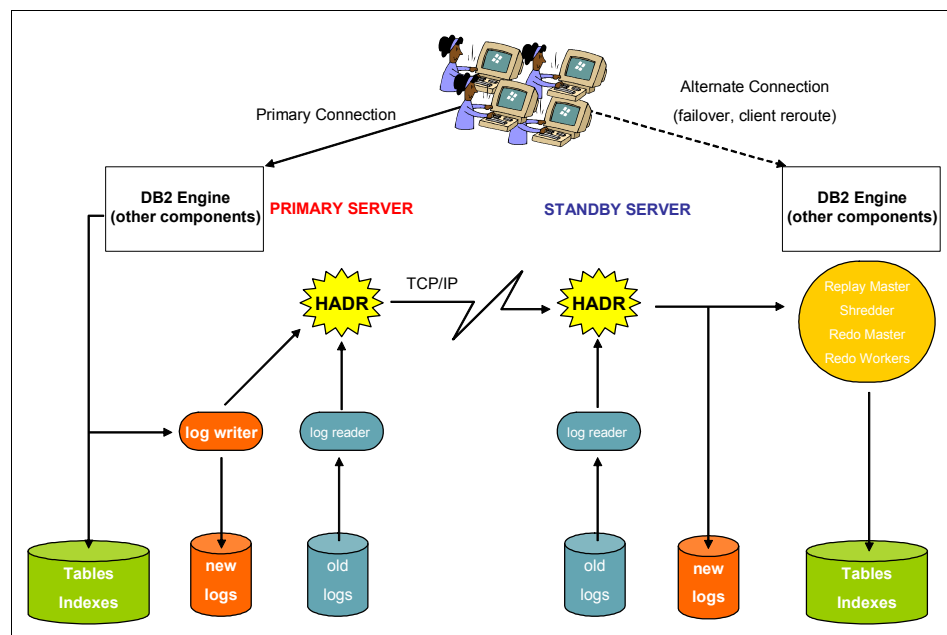


Figure 2-1 HADR overview

Using two dedicated TCP/IP communication ports and a heartbeat, the primary and the standby keep track of where they are processing currently, and the current state of replication; perhaps most importantly, whether the standby database is up to date with the primary. (known as HADR Peer state). When a log buffer is being written to disk on the primary, it is sent to HADR to be routed to the standby at the same time.

As mentioned, HADR communication between the primary and the standby is through TCP/IP, which enables the database to be replicated to a remote geographical site. This allows the database to be recovered either locally or at the remote site in case of a disaster at the primary database server site. The HADR solution thus provides both High Availability and Disaster Recoverability. As you can appreciate, HADR provides an incomparable improvement on conventional methods for DB2 disaster recovery, which otherwise could mean losses in terms of hours of committed transaction data.

If the primary system is not available, a takeover HADR by force operation converts the standby system to be the new primary system. After the maintenance or repair is done in the old primary system, you can bring the original primary DB2 up and return both DB2 servers to their primary and standby roles after reintegration of HADR, and after the original primary has caught up with any work that was done while it was unavailable.

In a DB2-ready environment, HADR is easy to set up and configure. The bottom line is that HADR is an efficient method of supporting high availability and disaster recovery.

## **Why HADR?**

The reasons to choose HADR as a solution are as follows:

- ▶ Ultra-fast failover capability
- ▶ Easy to set up and monitor
- ▶ Rolling upgrades without service interruption between non-major versions or changes requiring server recycling, and reduced change windows for other upgrades
- ▶ Transparent failover and failback for applications
- ▶ Built in clustering software in DB2 9.5
- ▶ Dramatically improved disaster recovery compared to conventional methods
- ▶ Negligible impact on performance

## 2.3 DB2 logical log architecture

In this section, we discuss the logical log architecture. This DB2 architectural background can be helpful for clearly comprehending the infrastructure behind HADR. We assume that the readers are familiar with the DB2 buffer pool architecture.

One of the main features that DB2 possesses to improve performance, and to guarantee the security and robustness of your data, is the capability of managing transaction integrity.

In order to preserve the logical integrity of the database, individual SQL operations are grouped together as a transaction. All operations inside a transaction are guaranteed to be executed in an all-or-none fashion - they are sometimes referred to as an atomic unit of work. If a failure occurs at any individual step, the transaction is aborted (rolled back) and all individual operations that have already been executed are undone. Transactions are also known as atomic, or logical units of work, in that while they can be constituted of smaller elements, they cannot be (and must not be) logically split up, either from the point of view of an application's specific business rules, or data integrity in general.

DB2's logical logging implements the Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) recovery algorithm, which is a Write-Ahead Logging (WAL) protocol to ensure atomic operations. In this protocol, every agent that modifies a page also generates a record of the operation known as a *logical log record*. This logical log record is stored in a special section of the disk known as a *logical log file*. Transactions have special logical log records that identify the beginning and the end of a transaction. In the event of a crash, the logical log files are read and all records in them are replayed. All records belonging to the incomplete transactions are rolled back, bringing the database to a consistent state.

ARIES is described in further detail at the following Web site:

[http://www.almaden.ibm.com/u/mohan/ARIES\\_Impact.html](http://www.almaden.ibm.com/u/mohan/ARIES_Impact.html)

For performance reasons, the logical log records are not written directly to the disk, but to a buffer area known as the *logical log buffer*. To minimize the amount of information lost in the unlikely event of a failure, the logical log buffer is flushed to disk every time an application commits a transaction.

For the ARIES protocol to work as designed, this condition must be met: No dirty pages in the buffer pool can be written to the disk (table space container) before the logical log buffer pages containing the logical log records that modified the buffer pool page are written to the logical log file. A dirty page is one that contains changed data or records which have not been committed yet — the data is in the middle of a transaction.

Figure 2-2 depicts a sequence of events that happen in the server when an application requests an update to a page in a table space. The db2agent is a process (in Windows platforms it is a Windows thread) that is in charge of executing the work requested by the application.

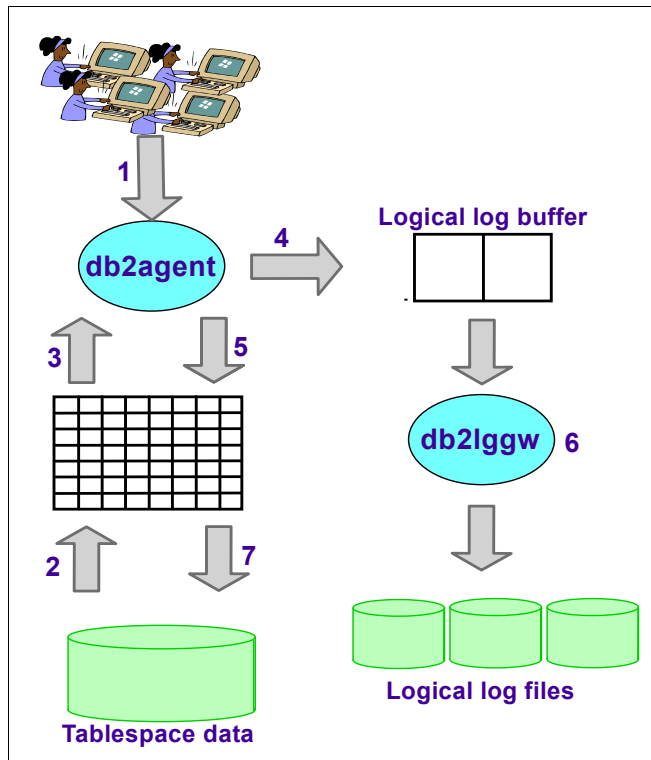


Figure 2-2 Application interaction with DB2 processes to update table space data

Here we describe each sequential event:

1. The application requests to update a row in a table.
2. The data is read (fetched) from the table space into a page in the buffer pool.
3. The db2agent receives the page from the buffer pool and modifies it.

4. The DB2 engine records what was changed in that page by writing a logical log record into the logical log buffer. This record can later be used to undo the operation in case of a rollback, or to replay the transaction in the event of a failure (rollforward).
5. The page is placed into the buffer pool. At this point it is not safe to write (externalize) the page from the buffer pool back into the table space, as the logical log record is still in the log buffer, but is not written to the log file yet. Because the log buffer only exists in memory, a system crash can cause the logical log record to be lost forever. If this change had been written to the table space before being recorded in the log file, a crash would result in DB2 not knowing about this transaction from the log files. DB2 would not be able to replicate that same data by means of a rollforward from the most recent backup image. Conversely, a crash after a successful write of log buffer data to the log file on the disk enables DB2 rollforward recovery to successfully replicate the data correctly in the table space container.

Note that because flushing the log buffer and the buffer pool are asynchronous operations accomplished by different engine dispatchable units (EDUs), db2agent is freed at this point to do any other work.

6. At some point an event triggers a logical log buffer to be written to the logical log files. This work is accomplished by an EDU called db2loggw. Events that might trigger the flush of the logical log buffer include these:
  - An application issues a commit and commit count is equal or higher than the value specified in the database configuration parameter MINCOMMIT.
  - A page in the buffer pool that references a log record in this buffer requires writing to the table space on disk.
7. After the logical record has been written to the logical log file, it is safe to flush the modified data page from the buffer pool into the table space.

Every operation executed by the database over a page in a table space generates a logical log record that is stored in a logical log file. Logging is made at the database level in DB2, so each database has their own log files and separate logging processes.

The logical log buffer is written to disk by the logger process db2loggw. There is another log EDU known as the *log reader*, which has the role of reading data from the logical log files during a rollforward operation, for example, or when a userexit application is invoked. The classical log reader used by DB2 version 8 is db2loggr.

With the introduction of HADR in DB2 V8.2, a new type of log reader architecture is created. This new architecture is implemented by an EDU known as *log file reader* (db2lfr). Presently, db2lfr and db2loggr coexist to manage different subsystems, but eventually, both of them are migrated to the new db2lfr EDU.



As of DB2 V9.5, both db2lfr and db2loggr still coexist, performing complementary roles, as described in the DB2 V9.5 Information Center:

<https://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0008930.html>

## 2.4 HADR architecture

HADR technology is tightly coupled to the DB2 logging and recovery strategy. The main design goal of DB2 HADR is to quickly failover the failed system to a standby server in case of a failure on the primary server.

The basic architecture of HADR is very simple. It is based on the shipping of log buffers from the primary database to a remote instance containing what is known as the “standby database”. Thus, in DB2 the HADR implementation allows for the database administrator (DBA) to decide which databases should be replicated.

After the appropriate start, HADR commands have been entered on both the standby and primary databases. The HADR system actually starts when the primary database becomes active.

In an HADR system, DB2 has two special EDUs to handle all HADR work. The one on the primary database is db2hadrp. Its counterpart on the standby database is db2hadrs. They are in charge of sending log buffers from the primary to the standby, processing system messages, and receiving logical logs on the standby system.

At database activation in the primary, the db2hadrp is spawned; this reads the database configuration file and open a port to listen for a connection from the standby. The relevant HADR database configuration parameters are described in 6.1, “DB2 HADR configuration parameters” on page 144. During this time no client connections are allowed. The primary waits for the number of seconds set in the configuration parameter HADR\_TIMEOUT for the standby connection, and then aborts the database activation if it is not able to successfully connect to an active standby database.

If the FORCE clause is used when issuing the START HADR command, the database does not wait for the standby to connect. The primary creates a listener and listens on the HADR port. The standby initiates to connect to the primary. If the standby's connection attempt fails, it retries connecting to the primary. DB2 clients are always able to connect to an active HADR primary database, and whenever the standby is ready, the actual HADR functionality becomes operative.

The downside of using the FORCE clause is that the database does not respect the setting of the AUTORESTART parameter and in the event of a crash occurring, crash recovery is performed irrespective of the current setting. Any other methods to start HADR, including ACTIVATE DATABASE and the first client connection, do respect the AUTORESTART settings.

After the connection is established, the status and “health” of the HADR system is monitored by a mechanism known as *heartbeats*. This is basically to verify that both the primary and the standby can see each other. Heartbeats are special short messages sent over the HADR connection from the primary to the standby, which are acknowledged and sent back by the standby. The heartbeat messages are spaced at known time intervals so each end can know how many heartbeats have been missed and take appropriate actions.

Upon successful connection of the standby database to the primary, the HADR system enters the *catchup phase*. During the catchup phase the size of the logical log gap from the standby to the primary is established and the primary starts sending all logical logs that are required for the standby to reach the same point as the primary.

The logical log reading is accomplished in the primary by the db2lfr EDU. This process relays logical log pages to the db2hadrp, that in turn relay the pages over the TCP/IP layer for the db2hadrs in the standby to catch and relay to its logical recovery subsystem.

After all the logs in the disk and the memory in the primary have been relayed to the standby, the HADR system enters the *Peer state*.

## **HADR topology**

The standard and—even in DB2 V9.5—the only currently supported topology for any one pair of databases is one primary and one standby database. This makes the most commonly seen layout as a pair of servers, one with a single DB2 instance with a single database acting as the primary, and the other with a single DB2 instance with a single database acting as standby.

However, this is by no means the limit to the ways in which multiple pairs of HADR databases can be implemented on multiple server nodes.

HADR replication takes place at a database level, not at the instance level. The standby server can have multiple databases from multiple primaries on it. Another configuration option is to implement two servers in an active/active mode cluster as shown in Figure 2-3, with HADR primary databases spread across both servers in a load sharing configuration. The advantage with this option is that it makes more efficient use of available CPU cycles, and other valuable server resources.

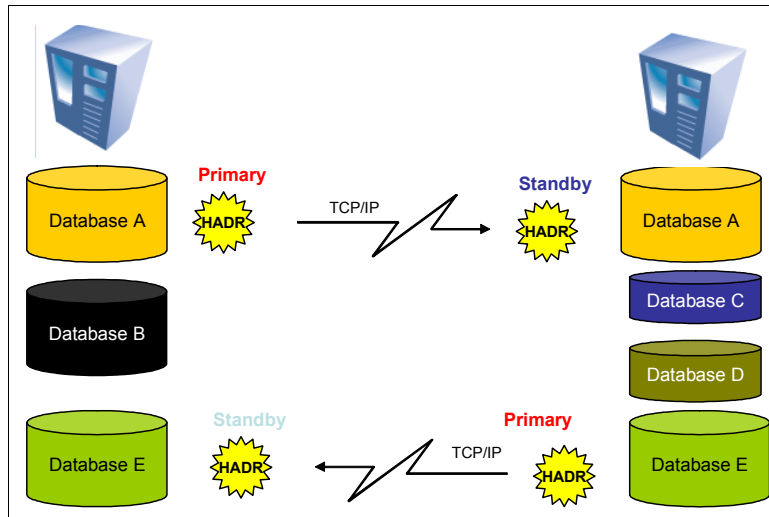


Figure 2-3 HADR takes place at database level not DB2 instance level

## HADR synchronization modes

With HADR, you can choose the level of protection you want from potential loss of data by specifying one of the three synchronization modes:

- ▶ Synchronous:
 

Log write is considered successful only when the log buffers have been written to the primary's log files and after an acknowledgement has been received that it has also been written to the standby's log files. There can be no data loss in this mode if HADR is in Peer state.
- ▶ Near-synchronous:
 

This is the default option. Log write is successful only when the primary's log buffer has been written to log files on the primary and an acknowledgement is received that the log buffer has been received on the standby.
- ▶ Asynchronous:
 

Log write is successful when logs have been written to the disk on the primary and log data has been sent through TCP/IP to the standby. Data loss can occur in this mode.

Figure 2-4 illustrates these HADR synchronization modes, and the point at which DB2 considers itself able to commit a transaction while in the PEER state.

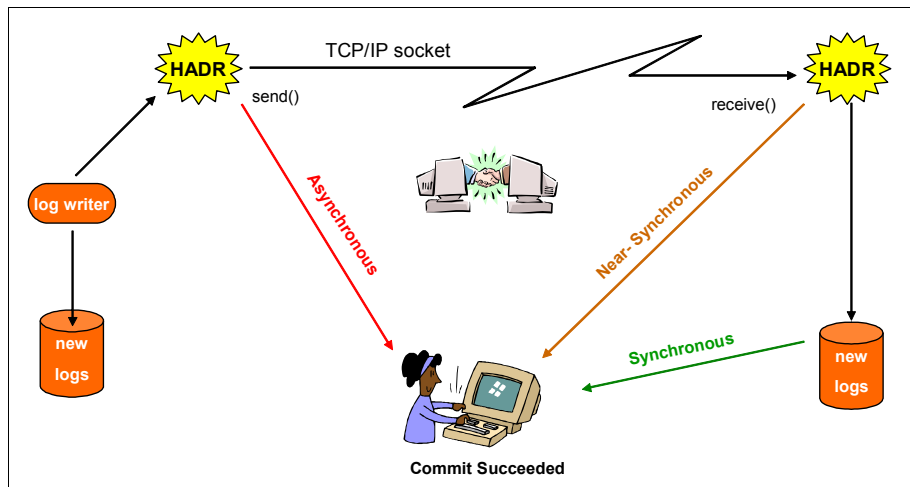


Figure 2-4 Synchronization modes

## 2.5 Terminology

Here we list and describe some of the more common terms used specifically with HADR. Most of these terms can be found in various DB2 manuals or the DB2 Information Center, where you can search and see them being used in context.

Generic definitions can be found in the DB2 9.5 Information Center Glossary:

<https://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.luw.glossary.doc/doc/glossary.html>

We recommend using the Search field in the DB2 Information Center to find in-context usage of these terms:

► Catchup phase:

HADR initialization always starts in the catchup phase, in which the standby tries to *catch up* to in-memory logs on the primary by replaying logs that have been written to the disk or the archive. During catchup, the standby can retrieve log files locally, or remotely from the primary through the HADR network connection.

► Failover:

In the context of HADR, this refers to changing the status of the standby in an HADR pair to become the primary, with full DB2 functionality, due to the original primary failing. The original primary can be brought up subsequently in the role of standby. Care must be taken to ensure that the original primary is truly non-functional at the time of failover. If both databases are functioning as primary, there is a conflict in the data update that HADR cannot resolve. The result is two incorrect versions of the database, which might be impossible to reconcile.

Failover is also often used in the context of a cluster, where resources on an active node which has failed, are made active on a previously passive or inactive node. This is nearly indistinguishable as a concept from Failover in HADR.

► Failback:

In the context of HADR, after a failover has occurred, which made the original standby into a primary, failback is the act of reverting this database back to a standby and bringing the original primary back as a primary once again. That means, switching the roles of the primary and the standby, while both are alive and healthy.

Failback is not a mandatory operation. A customer can choose to leave the databases in their reversed roles until another failover is necessary due to the failure of the new primary. This is especially likely in cases where the HA goal is ultra-fast failover, as failback can be viewed as a needless additional interruption in service.

Customers who choose this option must ensure that all DB2-related objects and processes, which HADR does not replicate, are the same on both the servers; especially things such as batch schedule tasks, which must be deactivated while the database is in the standby mode, and activated while it is in the primary mode.

In the context of a cluster, failback has effectively the same meaning as with HADR.

► Failure:

Failure is an event where any database service prerequisite component (DB2, operating system, server machine, or network) is no longer able to provide the service it is supposed to. HADR maintains data availability in the event of failure of any single component. If the failure affects only the standby system or the communication between the primary and the standby, data remains fully available on the primary without any disruption or required user action.

If the failure prevents the primary itself from providing DB2 functionality, the user can take the primary DB2 instance, the server, or both completely down (if it is not already) and initiate failover, which rapidly changes the standby system into a primary, making the data available again after only a brief outage.

Some failures can be detected and reported by HADR and DB2 to assist the user in deciding whether failover is needed or whether certain database components simply require attention. The HADR database processes report a failure when they are unable to communicate. The DB2 database processes report a failure when they detect a fault. These are the only DB2 database components that report failures as provided by the DB2 software.

► Out of band:

In this book, out of band refers to operations that can be performed on the primary database, and which must be replicated at the standby, but cannot be repeated solely based on log records. Such operations are non-logged or not completely logged at the primary. In order to replicate such an operation, its results (a copy of the new or modified object, for example) must be passed to the standby by some other means. This “other means” is considered to be *out of band* with respect to the log stream. Note that this use of the term is different from the meaning of *out of band* in TCP/IP. Thus, it is preferable to refer to the specific operations or to *non-logged* or *not completely logged* operations to avoid any confusion.

► Outage period:

The outage period is the amount of time it takes to failover DB2 functionality from a broken DB2 primary database to the standby. This period of time starts from the point at which failover is initiated to the point that DB2 functionality is restored. To a client connecting to the DB2 database, it is seen as a connection failure, at which point retry logic should be working to ensure that the transaction is not lost.

► Peer state:

After the standby catches up with in-memory logs on the primary, HADR enters the Peer state, in which the primary ships the log page to the standby whenever it flushes a log page to the disk. The log pages are replayed on the standby as they arrive. The pages are also written to local log files on the standby so that the primary and the standby have identical log file sequences. Transaction commits on the primary wait for acknowledgment messages from the standby or at least for a successful return of log send calls, depending on the user-specified log shipping mode level.

- ▶ **Primary (database):**

This is the principal (master) copy of the database. Applications apply updates to the primary database and those updates are propagated to the standby server via log shipping.
- ▶ **Primary reintegration:**

In some cases, the old primary database can rejoin the HADR pair after a failover. Because the old standby is now the primary, the old primary can only join as a standby. This is called primary reintegration. After reintegration, a failback can optionally be performed.
- ▶ **Standby (database):**

This is a copy of the primary database. It is not updated directly by the application. All updates occur by rolling forward log data generated on the primary database.
- ▶ **Split brain:**

This refers to the condition of having both databases in an HADR pair acting as primaries simultaneously. Such a situation leads to the databases becoming inconsistent with each other and should be avoided at all costs. HADR never automatically makes state transitions that result in two primaries, and when the pair can communicate with each other (that is, the network connection between them is operating properly), they also prevent a user from creating this situation in an unforced takeover. HADR also does not allow an inactive primary database to be activated without a successful connection to a standby within the HADR\_TIMEOUT period.

However, split brain is still possible in a situation where the user instructs the current standby to become a primary while the communications link between the pair is down, if either the current primary is still active or if it is brought back to the non-HADR, or standard mode. But, this is a *manual* command that runs only in the most dire cases.
- ▶ **Standard (database):**

In the context of HADR, *standard* means a normally operating non-HADR database. That is, a database not using the HADR feature, and therefore not operating in either the primary or the standby mode.

▶ Synchronous mode:

In the Peer state, the primary does not consider a transaction as committed until it gets an acknowledgment message from the standby confirming that the relevant log data has been received and written to the disk on the standby. Therefore, if a transaction is committed on the primary, it is guaranteed to be persistently stored in the standby's log file. Even if the standby crashes before it is able to replay the log, it can still replay it from its own log file when it restarts. There is no transaction loss in a Synchronous mode failover as long as the primary was in Peer state at the time of the failure.

▶ Near-synchronous mode:

In the Peer state, the primary does not consider a transaction as committed until it gets an acknowledgment message from the standby confirming that the relevant log data has been received and written to the main-memory of the standby.

▶ Asynchronous mode:

In the Peer state, the primary does not consider a transaction as committed until it successfully submits the relevant log data to the network. The primary does not wait for any acknowledgment message that the log data was received.

▶ Takeover:

Takeover is the act of the HADR standby taking over control of the database from the old primary server and becoming the new HADR primary. Takeover is always initiated from the standby. If the primary can be reached over the network as in an unforced takeover, the standby asks it to switch to standby, performing cooperative role switching. Otherwise, the standby takes action unilaterally (with the risk of dual primary/split brain). Failover is a unilateral takeover. Failback is a form of cooperative role switching performed after first reintegrating the repaired server as a standby. Note that HADR *takeover* can be performed outside the context of failover and failback. A user might want to switch HADR roles for rolling upgrade, which does not involve a failure at all.

▶ Engine dispatchable unit (EDU):

The engine dispatchable unit (EDU) can be implemented either as a process or a thread, depending on the platform architecture. Note that in DB2 9.5, threading has been implemented on all the platforms.





## HADR setup

In this chapter we describe the steps to set up a High Availability Disaster Recovery (HADR) environment using both the command line and Graphical User Interface (GUI). We cover the basic operations, including START HADR, STOP HADR, and TAKEOVER HADR commands. We also provide troubleshooting tips.

## 3.1 Requirements for setting up HADR

Before you start your setup of HADR, first you must know the requirements for setup. As part of this book, we provide a basic outline of what you require to get started (for more details, refer to *Data Recovery and High Availability Guide and Reference*, SC23-5848). Along with the requirements, we have also added our recommendations.

### Requirements

Setting up an HADR includes the following requirements:

- ▶ HADR is a DB2 optional licensing feature. The full use of HADR is allowed with Enterprise Server Edition (ESE), but you might need more DB2 licenses for the standby server(s). Consult with IBM IM Sales representatives for the details.
- ▶ The operating system on the primary and standby databases should be the same version, including the patches. For a short time, during upgrades, they can be different.
- ▶ A TCP/IP interface must be available between the HADR primary and the standby.
- ▶ The DB2 version and level must be the same on both the primary and the standby systems.
- ▶ The DB2 software for both the primary and the standby database must have the same bit size (32-bit or 64-bit).
- ▶ Buffer pool sizes on the primary and the standby should be the same. Note that if you build the standby database by restoring the database using the backup copy from the primary, the buffer pool sizes are the same, because this information is included in the database backup.
- ▶ The primary and standby databases must have the same database name. This means that they must be in different instances.
- ▶ Table spaces must be identical on the primary and standby databases including:
  - Table space type
  - Table space size
  - Container path
  - Container size
  - Container file type
- ▶ The amount of space allocated for log files should also be the same on both the primary and standby databases.

## Highly recommended parameters

These are some of the highly recommended parameters to set up an HADR:

- ▶ Use identical host computers for the HADR primary and standby database servers. That is, they should be from the same vendor and have the same architecture.
- ▶ Use a high-speed, high-capacity network for the TCP/IP interface.
- ▶ Ensure that the primary and standby database servers have equal amounts of memory.
- ▶ Have identical database (DB) and database management (DBM) configuration parameters.

## 3.2 Setup and configuration

Here we perform the steps to set up HADR on an existing installation of DB2. Our Lab example uses 32-bit DB2 9 for SuSE 10 Linux on Intel® (kernel 2.6). However, these steps are applicable to all candidate environments.

### 3.2.1 Preparing the environment

In order to prepare a real world pre-change window environment to set up HADR, there are a few things you can plan for, which allow you to be ready well beforehand. Unless you are currently still using circular logging, the only downtime required for setting up HADR is for updating database configuration parameters. This can be avoided by schedule the work in a regular maintenance window. There should be no pressure to minimize your change window, other than a recommendation to avoid periods of heavy logging if you want to avoid a long wait for the standby to catch up before reaching the Peer state.

If you are still using circular logging in your real world environment, you might want to set aside a separate change window before commencing HADR setup to plan for the changeover to archive logging, and all the associated considerations for log archival, overflow, housekeeping, log file size, and retry on failure. If you are testing HADR setup on a sandbox environment, this is only a minor consideration, and no separate planning is required.

As basic components for this Lab, you require two DB2 instances of the same FixPak level. You can expect to see something like the following text in a pop-up message if your instances do not match:

```
The selected standby system is not at the same DB2 version as the
primary system. HADR requires identical DB2 version on the primary and
```

standby systems. MYSERVER - DB2 - DB2Version(8,2,6). LEAD - DB2\_LEAD (db2inst1) - DB2Version(9,1,0)

HADR works even if both instances are located on the same server (or LAPRs), but a real-world environment has separate servers, as we have used in our example.

Ascertain whether you intend to use the GUI HADR Setup wizard or the DB2 *Command Line Processor* (CLP) to configure HADR. The GUI wizard is perfect for introducing people who are new to the concept of HADR, but it has all the complexity and high network bandwidth overheads of a GUI. Ensure that you can get a fast and reliable display of the DB2 GUI interface. This is even more important if you are intending to use an X11 emulator for remote setup of a Linux/UNIX system.

The DB2 CLP environment is suitable for experienced administrators or for repeated configurations on multiple HADR server-pairs. It is also suitable for environments where it is impractical to get a GUI interface working, whether locally or remotely, such as AIX with telnet-only access, or systems with no Java™, or systems with limited administrative network bandwidth.

If you work in a high-security environment where people performing DB2 installs and maintenance are not given `sesu/sudo root` or Windows Administrator authority other than in a special change window, take the necessary steps to request that authority.

If you are running HADR setup from the GUI wizard as the DB2 instance ID for example, this specifically requires that the DB2 instance ID has the authority to update the `/etc/services` file (`%Systemroot%\system32\drivers\etc\services` on Windows), unless these ports have already been registered. You might also have to manually update the `/etc/hosts` file if you do not have a reliable Domain Name System (DNS) in place, but still want to use host names rather than IP addresses.

The two servers you would use in a real world implementation must be connected by a very reliable TCP/IP connection. They must preferably have no firewall between them, or at least with port 523 open for the DB2 Administration Server, the main port for each DB2 instance, and the two ports you set aside for HADR communication must be left open.

Depending on your environment, and your choice of HADR SYNCMODE, you might be working with physically separate servers:

- ▶ Over an intranet WAN (recommend ASYNC or NEARSYNC; more for disaster recovery functionality as opposed to a strictly high availability implementation)

- ▶ On a LAN (recommend NEARSYNC)
- ▶ On servers sitting physically next to each other in the same rack (recommend SYNC)

Our example uses the latter configuration.

If your environment is not tolerant of performing frequent backups, even if they are online, then take whatever steps are necessary to get a window in which to perform one as a starting point for the HADR change window.

If you want to reduce the number of steps performed inside the HADR GUI wizard as much as possible, in addition to changing from circular to linear/archival logging, you can catalog entries for the remote system on both the servers beforehand:

- ▶ ADMIN TCPIP NODE for the DB2 Admin Server
- ▶ TCPIP NODE for the DB2 instance
- ▶ DATABASE alias for the DB2 database

Example 3-1 and Example 3-2 show the commands to catalog the database server and database.

*Example 3-1 Commands to catalog remote entries on primary server (LEAD)*

---

```
db2 catalog admin tcpip node POLONIUM remote x.x.x.x remote_instance
db2inst1 system POLONIUM
db2 catalog tcpip node DB2_POL remote x.x.x.x server 50001
remote_instance db2inst1 system POLONIUM
db2 catalog database SAMPLE as SAM_POL at node DB2_POL
```

---

*Example 3-2 Commands to catalog remote entries on standby server (POLONIUM)*

---

```
db2 catalog admin tcpip node LEAD remote x.x.x.x remote_instance
db2inst1 system LEAD
db2 catalog tcpip node DB2_LEAD remote x.x.x.x server 50001
remote_instance db2inst1 system LEAD
db2 catalog database SAMPLE as SAM_LEAD at node DB2_LEAD
```

---

One final (and optional) consideration concerns the host and port entries for each server. The HADR GUI wizard attempts to add two port entries for HADR into `/etc/services`, but you can arrange to have them added beforehand, especially if your local operating system security administrator does not give the DB2 instance ID enough authority to do so. Remember that if you do this, you are required to override the default entries that appear in the wizard, as it tries to auto-increment the port numbers to something that does not already exist in the `/etc/services` file.

For our Lab example, we use HADR ports 55001 for LEAD, and 55002 for POLONIUM. You can also arrange to have the unqualified remote host name registered in /etc/hosts file on both the servers, so that you do not have to rely on IP addresses. Because they are not generally dynamic values, it is acceptable in the wizard to enter the physical port number for the local and remote DB2 instances rather than the service name alias (port 50001 for both instances in our Lab). For administrative and documentation reasons, it is still important to register the port numbers for the local services in each server's /etc/services file.

Having completed this, you are now ready to perform an HADR setup.

### 3.2.2 Configuration using the Setup HADR wizard

Figure 3-1 illustrates the basic intended high-level architecture of our Lab environment.

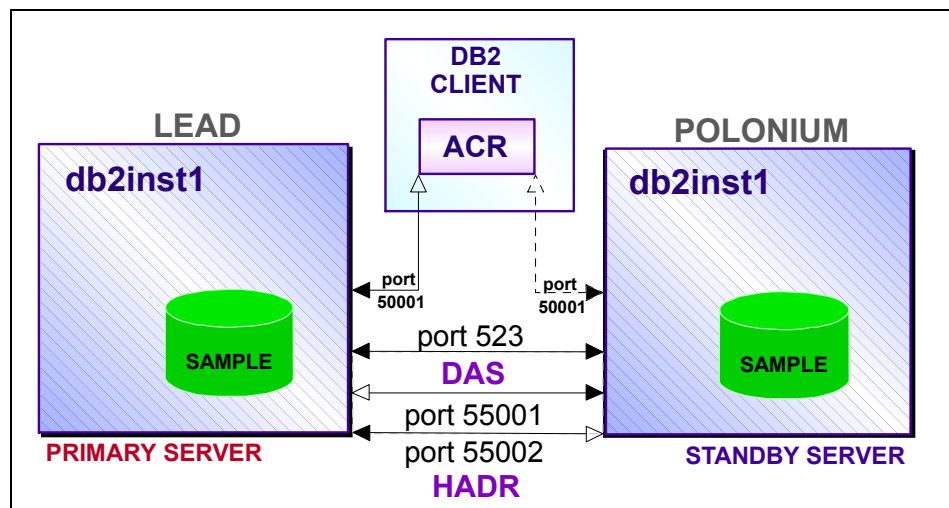


Figure 3-1 Lab environment high-level overview

In Figure 3-1, the two example instances are both called DB2INST1 on server host names LEAD and POLONIUM, in a non-clustered configuration, with basic TCP/IP ethernet network adapter connectivity. LEAD is initially set up as the HADR primary, and POLONIUM is initially the HADR standby.

The ACR box refers to *Automatic Client Reroute*, which is built into the DB2 Client and configured on the DB2 server. After being configured, ARC is there to route the client connections to whichever DB2 instance is currently set as the primary. For more details about ACR, refer to Chapter 5, “Automatic client reroute” on page 121.

In our Lab, clients can communicate with the DB2 instance through port 50001, and DB2 HADR communicates between the servers through ports 55001 and 55002. The DB2 Admin Server (DAS) GUI interface communicates through port 523.

The database being set up in HADR configuration is called SAMPLE; we eventually give this database an alias on each respectively remote server, so that it is accessible from the GUI Control Center Manage HADR dialog box, and for general remote takeover, stop, and start commands.

In the Lab, we start out with only one SAMPLE database, sitting in the DB2INST1 instance on one of the servers (LEAD).

We finish with SAMPLE acting as HADR primary on the DB2INST1 instance on LEAD, and as HADR standby in the DB2INST1 instance on POLONIUM.

In the step descriptions that follow, the term *remote* is used in the context of the server that we eventually configure the *standby* database on, and likewise, the term *local* is used interchangeably with *primary*. This is simply because we are running the wizard locally on the server that becomes the primary. Initially, we do not have a primary or a standby, just the local server where the setup GUI dialog boxes appear (LEAD for our example) and the remote server that gets a database restored on it (POLONIUM).

After configuration is complete, the primary and standby roles can be swapped if you want, so these are temporary descriptions at best. To help with establishing a more permanent frame of reference, we have included our Lab example server names (LEAD and POLONIUM) in parentheses when listing a given server, so you can substitute your own server names for each situation.

## Beginning HADR configuration

For beginners, the most user-friendly way to achieve a working HADR environment is to use the GUI wizard.

1. Start the DB2 Control Center (db2cc), and expand the **All Databases** tree on the left pane to show your databases.
2. As shown in Figure 3-2, right-click the database you wish to set up, and click **High Availability Disaster Recovery** → **Set Up ...**

**Note:** You might be prompted to enter a user ID and password here. This should be any user ID that has DB2 SYSADM authority on the local/primary instance, DB2INST1 in our example.

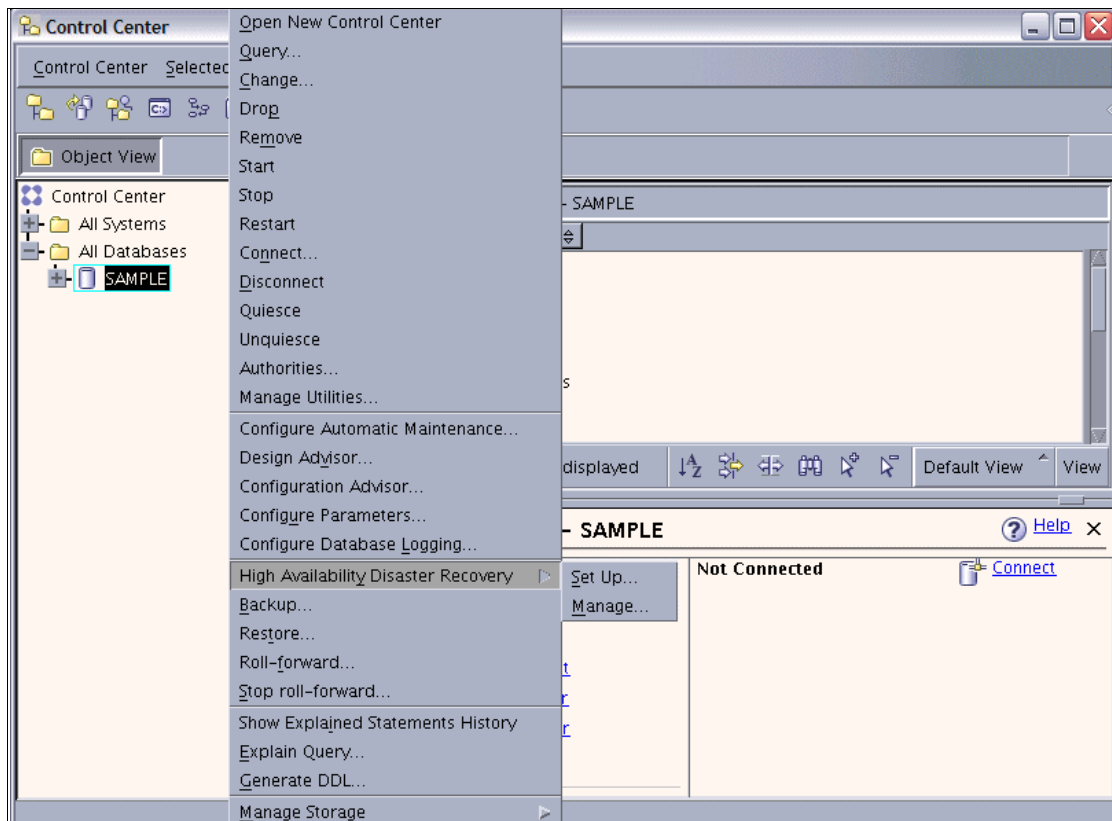


Figure 3-2 Starting HADR Setup wizard

3. The next screen is an introduction, giving you notes to keep in mind. Click **Next** after reading them briefly.



4. *Confirm the primary database selection* (see Figure 3-3). This points out that the database must use linear (non-circular) logging, and must not be in a partitioned environment.

If the database is still using circular logging, click **Configure...**, and follow the subsequent wizard steps to configure linear/archival logging.

If the database is already in the linear or archive logging mode, you can skip the next several sub-steps (step a through step k) and go straight to step 5.

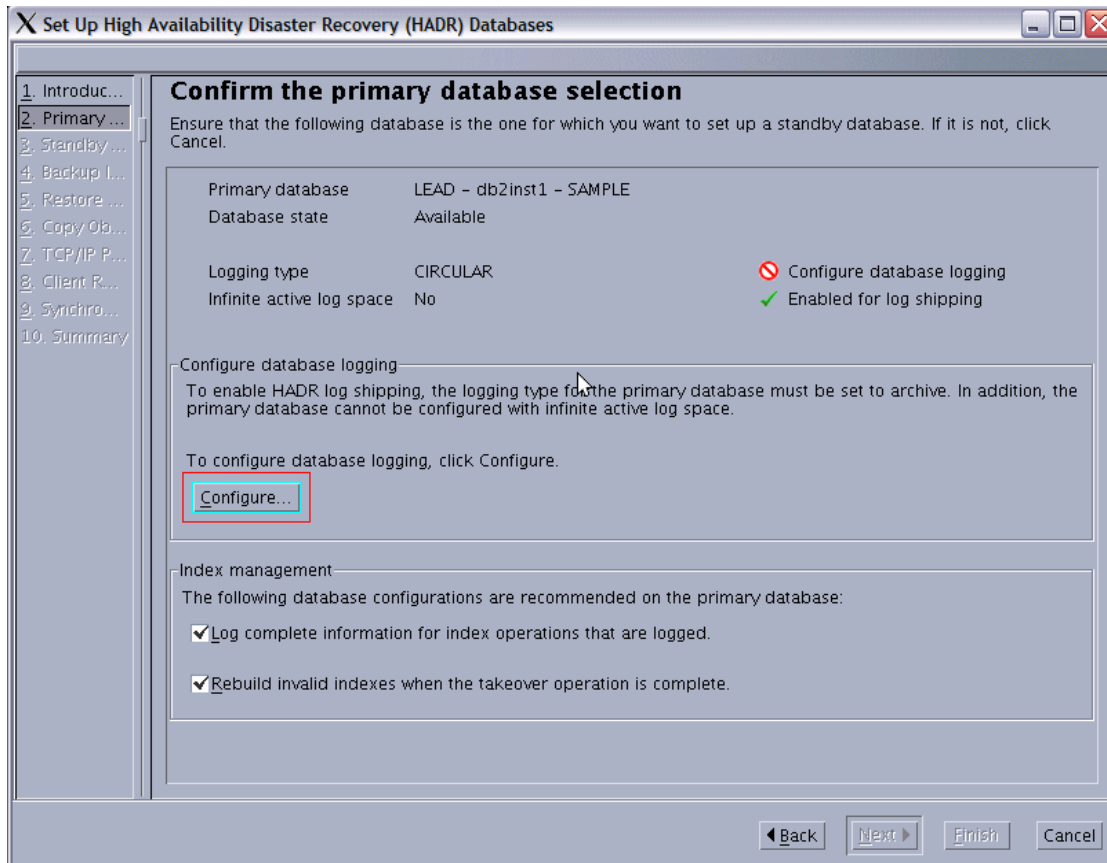


Figure 3-3 *Confirming the database to use as primary, and configure it to not use circular logging*

**Note:** Ignore the “inconsistent” state reference. Databases that are active are usually inconsistent. It just means that not everything has been written from log buffers to disk yet, or a pending state, or uncommitted activity.

- a. If the database is in the circular logging mode, the first dialog box that opens after you click **Configure** is *Choose your logging type*. Select **Archive logging**. Click **Next**.
  - b. *Choose how you would like to handle your archived logs*: We would generally select **Use DB2 to automatically archive the log files**, and then specify a location such as `/usr/db2/archlog` (\*nix) or `C:\db2archlog` (Windows). Click **Next**.
  - c. *Choose the number and size of your log files*: We accept defaults here, but refer you to *DB2 9.1 Administration Guide: Planning*, SC10-4223, Chapter 5 for recommendations based on your requirements. Click **Next**.
  - d. *Specify the location of your log files*: We would specify a location such as `/usr/db2/actlog` (\*nix) or `C:\db2actlog` (Windows). Click **Next**.
  - e. *Specify where to store your backup image*: This is required as part of the conversion to archive/linear logging, as the database is placed in backup pending after changing critical database configuration parameters, and logging recommences in the new linear format from that point in time. We would specify a location such as `/usr/db2/backup` (\*nix) or `C:\db2backup` (Windows). Click **Next**.
  - f. *Specify options for the backup*: Accept the defaults, compress the image if you want to save time transferring the file to the standby server later on. Click **Next**.
  - g. *Review the actions that take place when you click Finish*: This is just a summary of the steps that occur. You can see the actual commands by clicking **Show Command**. Click **Finish** or click **Back** to fix any incorrect settings as necessary.
  - h. A pop-up message is displayed, warning you that the database is going to be taken offline for a full backup. Click **OK**.
  - i. The DB2 Progress pop-up goes through the motions here. Wait for it to disappear.
  - j. The DB2 Message pop-up gives you a result summary. This should be successful, Click **Close**.
  - k. Now we are finally back at the dialog box we left in step 4, and can proceed. Click **Next**.
5. *Identify a standby database*: After your database logging is in a state ready to start HADR configuration, and has been deactivated, you can proceed with the next step, shown in Figure 3-4. This dialog box is displayed irrespective of whether you already have a database sitting on another server. All you need at this stage is to have a DB2 instance running there.

The wizard knows whether the database has been successfully configured behind the scenes, and it refreshes the status to allow you to proceed. You might be asked for the DB2 Administration Server user ID and password in order for DB2 to connect to the standby server, copy backup files, and so on.

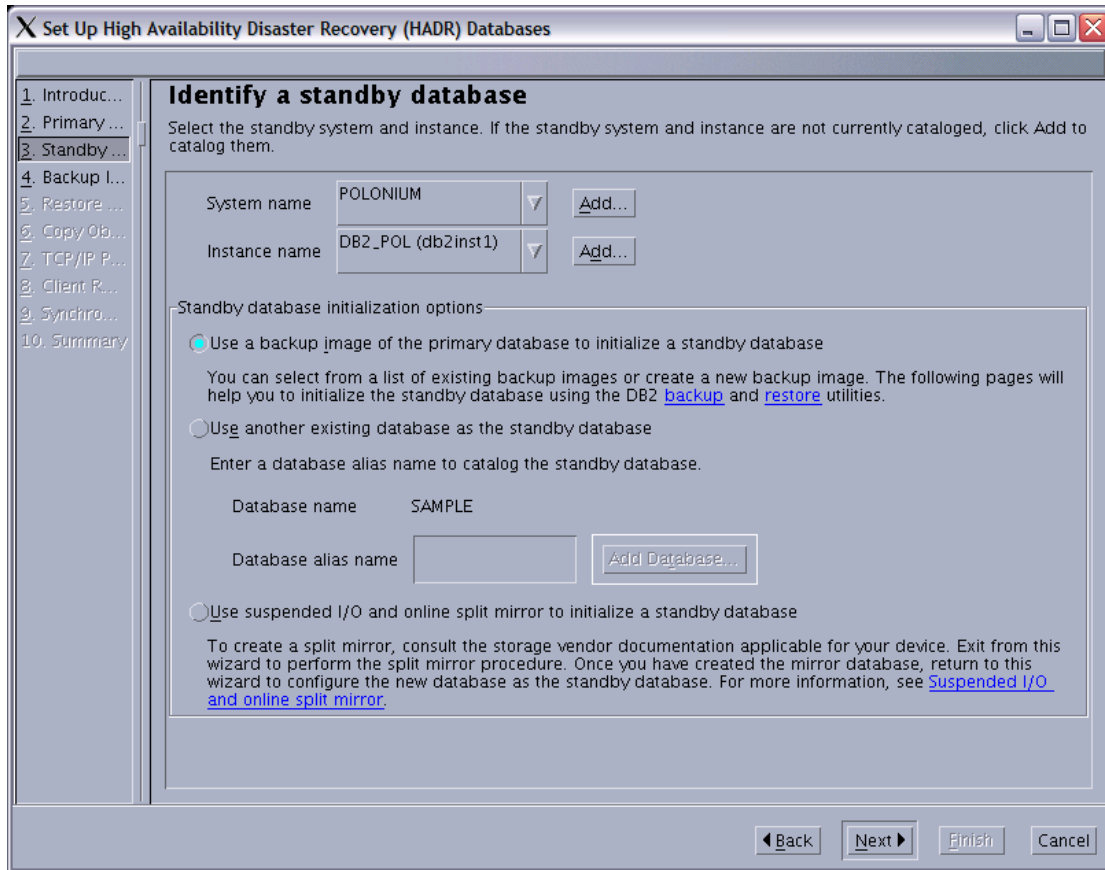


Figure 3-4 Choosing a database to use as standby

- a. If you have not already done so, the wizard prompts you to catalog a standby server IP/host name, Instance node name, and to either specify a backup image off the primary server, or an existing remote database. In order to have discovery working, the DB2 Administration Server must be running on the standby server (in our example, on POLONIUM).

Figure 3-5 shows the second dialog box in this series asking for an *Instance node name* to use for the remote instance, host address, and port number. The first dialog box is very similar, asking for a database alias to use for the remote database.

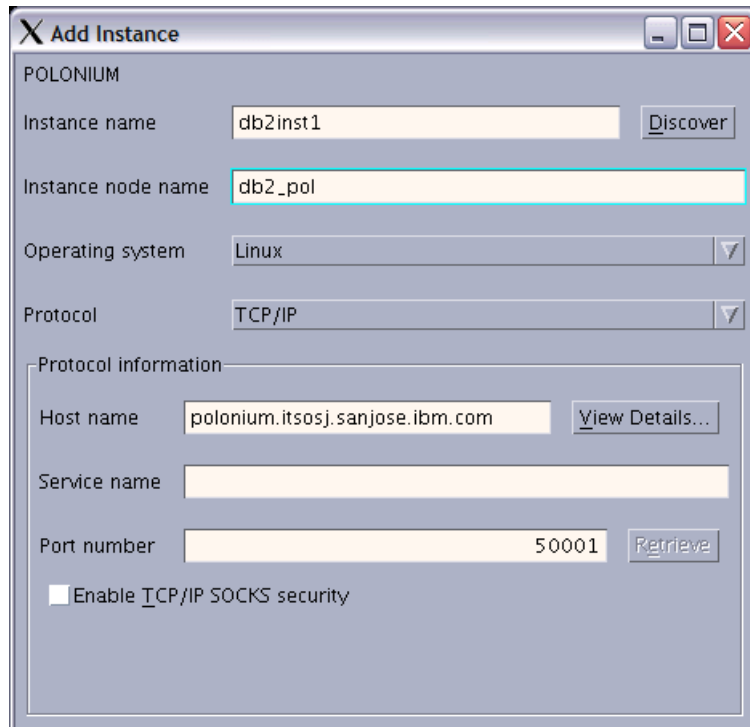


Figure 3-5 Catalog an Instance node name for the remote server's DB2 instance

As a brief explanation of these sub-steps, when you click **Add...** against *System name* and later against *Instance name* in Figure 3-4 on page 57, you can find yourself adding the host name/IP address for the remote *System* and assigning a node name, and then adding the host name/IP address for the remote *Instance* and being asked again to assign a node name. If you specify the same node name, the wizard reports that a node with that name already exists, and fails with an SQL1018N error.

Different node names must be specified for the System (the name you want the Control Center to display for the server/host name), and for the Instance (the name you want the Control Center to display for the instance).

In the *Add System* dialog box, the three mandatory fields for details of the remote server (System, Host, and Node name) are described as follows:

- **System name:** If you do not have a working DNS and you use the IP address of the server in the next field, put the host name value here. By default, entering a value in this field first fills out all the three fields with the same value, which is usually acceptable in a DNS environment with good naming standards.
- **Host name:** Either the IP address, the DNS, or the host name of the server registered in the `/etc/hosts` file.
- **Node name:** This can be any user-assigned value. Within DB2, it is a nickname, which represents the server, as opposed to the *Instance node name*, which represents the instance. In Windows, this is meant to refer to the NetBIOS Workstation name (dbm cfg parm: NNAME). There should be a hover/pop-up with this descriptive instruction if you hold the cursor on the node field. Practically speaking, it represents an ADMIN TCPIP NODE, which you can get a current list of with the following command:

```
db2 list admin node directory
```

The remote server's dbm cfg might not even have a NetBIOS name defined, so you can just use any appropriate name up to eight characters.

The fact that you are required to catalog the *Admin TCPIP Node* partially explains why you must have the DB2 Admin Server running at both ends in order to set up and manage HADR using the GUI interfaces and wizards.

When you are finished with the Add System dialog box, click **OK**. You might be prompted at this point by the DB2 Admin Server to enter a SYSADM authority user ID and password for connectivity to that new system entry. If you receive a communication failure after correctly entering a valid user ID and password in this dialog box, it could be related to the firewall for port 523, but it is best to check through all the possibilities described in the troubleshooting guide presented in 3.4, "Troubleshooting" on page 86.

If you receive the SQL1018N error message due to a duplicate node name, and you want to remove the incorrect entry and use values in your Add System dialog box without cancelling the Setup HADR wizard, you can open a separate Command Window, and enter the following command:

```
db2 uncatalog admin node <nodename>
```

In the *Add Instance* dialog box, the *Instance node name* is the nickname you create for the remote instance, cataloged for use on your local system (this is what gets displayed in the Control Center).

Remember, the System node name and the Instance node name are not meant to be the same. Behind the scenes, the System node definition is cataloging an ADMIN TCPIP NODE entry; the Instance node definition is cataloging a regular TCPIP NODE entry. That is why they are two separate layers in the DB2 Control Center tree structure.

One tip to avoid conflicts, while inside the wizard, is to add the system and instance definitions beforehand. This can be done with the Control Center or the *Configuration Assistant*. However, if discovery of the remote objects is not working for whatever reason, you have to catalog the remote server, node, and database alias manually; see Example 3-1 and Example 3-2 on page 51.

If you cataloged the ADMIN TCPIP node with a system name (for example, Figure 3-5 on page 58 shows the system name value of `polonium.itsosj.sanjose.ibm.com`), you might have to insert the TCP/IP address and host name entry for the remote host/server (POLONIUM) in your local `/etc/hosts` file. For Windows, this address and host name would have to go in the file: `%Systemdir%\system32\drivers\etc\hosts`. Otherwise, you can get TCP/IP errors such as those shown in Figure 3-3.

*Example 3-3 No local entry in /etc/hosts for the remote hostname*

---

```
SQL22212N  A DB2 Administration Server communication error has
been detected.  Client system: "x.x.x.x".  Server system
"x.x.x.x".
```

---

This is because the DB2 Admin Server on the remote system might have the host/server name down in text rather than a literal IP address, and that is what DB2 on the local host tries to resolve, even when you have specified literal IP addresses for your remote System and Host names. Note that you can receive the SQL22212N error for other more generic reasons, listed in the pop-up message at the time you receive the error message.

For the fields specifying the service name and the port number, we have used port 50001 for both DB2INST1 on the primary and DB2INST1 on the standby — that is, the primary DB2 port for the DB2 service for the instance on each server; this is the port with which clients connect to DB2.

HADR itself uses two different ports, one for each server. We use 55001 and 55002; these ports are for log transmission and HADR administrative traffic.

After having avoided or corrected the foregoing communication issues, a prompt should appear for a user ID and password to connect to the remote instance; use any SYSADM user ID from the remote instance (POLONIUM). The pop-up message shows an error code SQL22201N with reason code “1”, but that is expected, because you have not entered the DB2 user ID or password yet.

- b. The *Standby database initialization options* section in Figure 3-4 on page 57 gives you three options to initialize the database on the standby system. The first one (*use a backup image of the primary database to initialize a standby database*) should give a simple and successful HADR setup. That is what we use.
6. *Specify a backup image of the primary database* (see Figure 3-6): If you have just set up your database for linear logging in this wizard, you must have created an offline backup, which is perfect for use here in HADR configuration.

If you were already using linear/archival logging, this is the point at which you choose **Select a backup image from the list provided** and choose the latest backup, or if the backups listed are not recent, run an online backup now with **Backup the primary database**.

The wizard is able to use DB2 ports to file transfer the backup on the local system across to the remote system. If the backup image that you intend to restore from is not listed here — for example, if the backup image is being transferred to the remote server over ftp or through filesharing — then click **Enter the backup image information** and specify the subdirectory (DB2 Admin Server is capable of navigating the file system of the remote server and also the local server), and the date and time for the backup image you are using, as found in the backup file name.

If you use this method, click **Next**, and you are presented with the restore dialog as shown in step 7. This appears similar to Figure 3-7 on page 63 but without having to copy the backup image across if you specified the location as being on the remote server.

In our example, we choose **Select a backup image from the list provided** as shown in Figure 3-6 and select the latest backup.

Click **Next**.

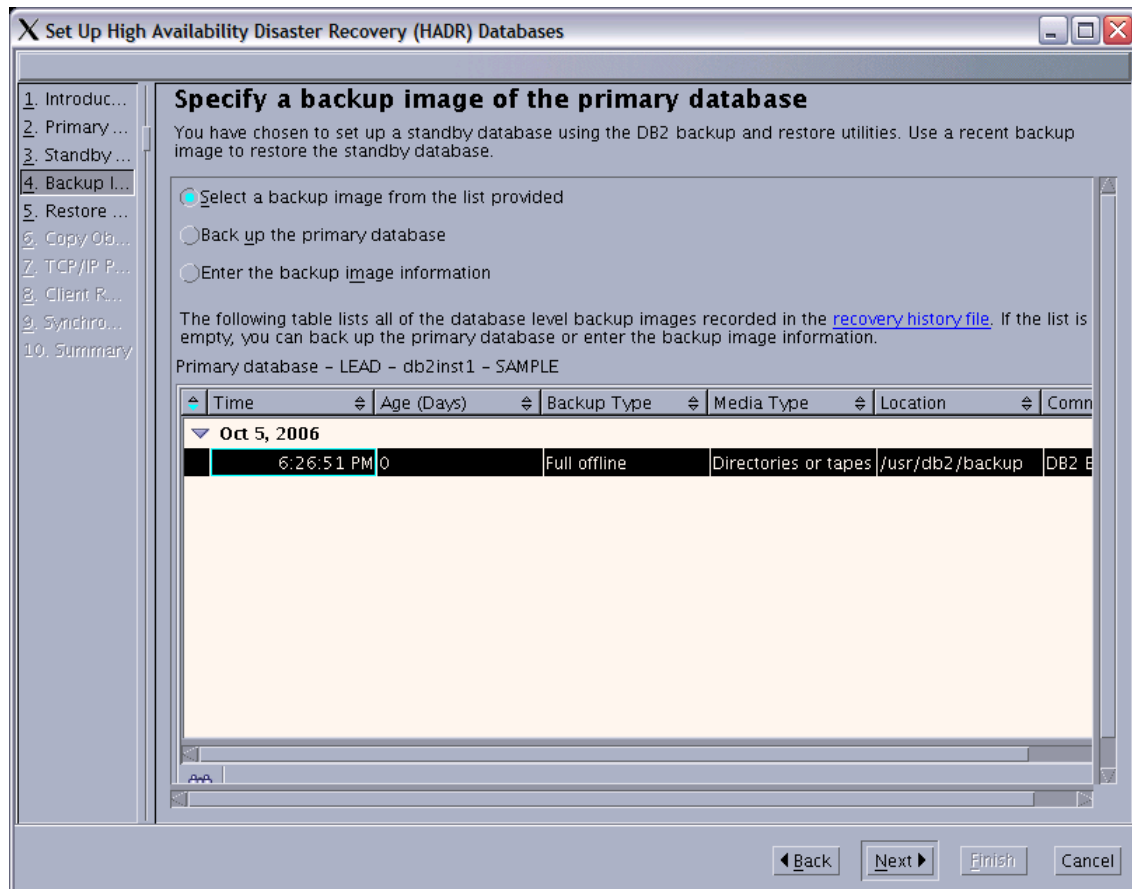


Figure 3-6 Selecting the preferred primary backup image to use for a restore on the standby server



7. *Restore database on standby system*: In Figure 3-7, specify an alias for the standby database on the remote system (POLONIUM) to be used by the primary instance on the local system (LEAD). This should exist from the time the standby database was identified in Figure 3-4.

For this example, we test the ability of DB2 file transfer by selecting **Copy the backup image from the primary system to the standby system**. Specify the location on the standby system where the backup file is to be transferred by the wizard. Click **Next**.

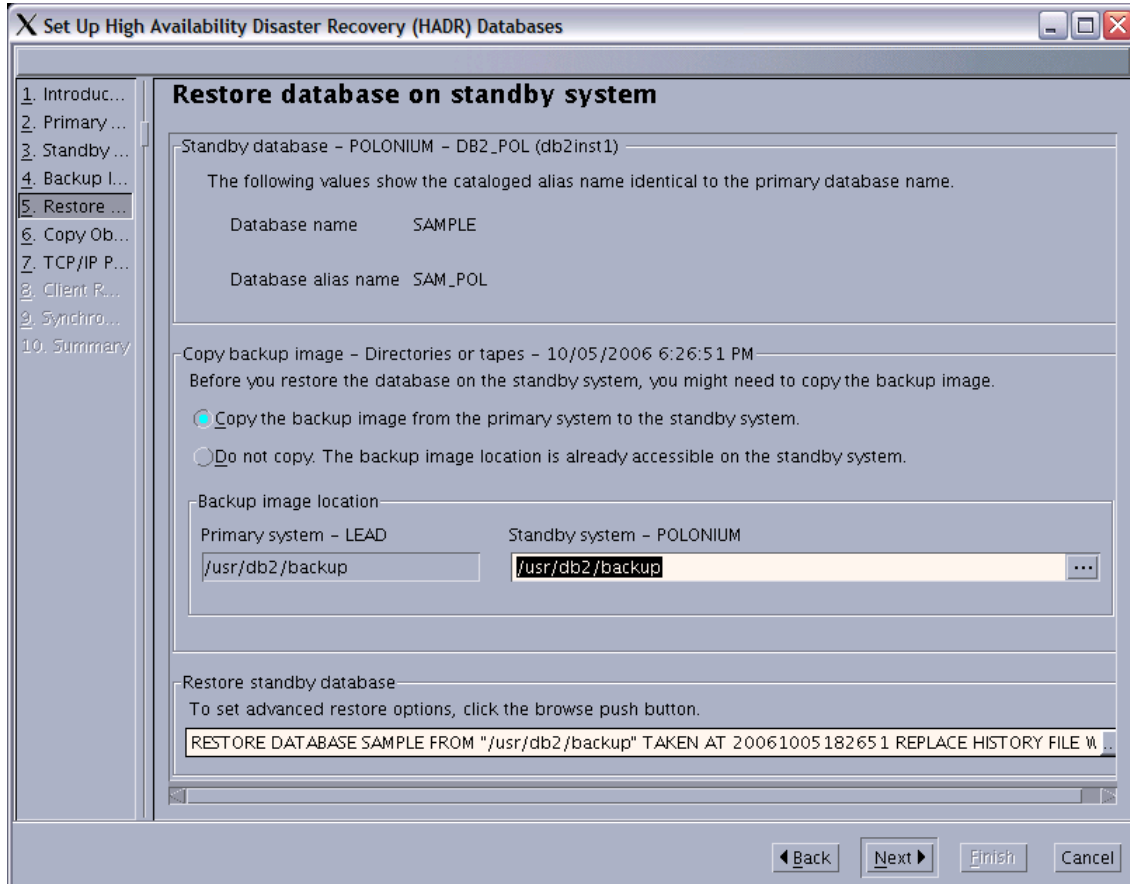


Figure 3-7 Restoring the primary backup image across to the standby server

8. *Copy objects to the standby system* (see Figure 3-8): If you have DB2 objects that do not get copied inside a database backup file, such as external code for UDFs and stored procedures, you can identify those objects here and have DB2 move them for you. Because we had none of those, we left the dialog box blank. Click **Next**.

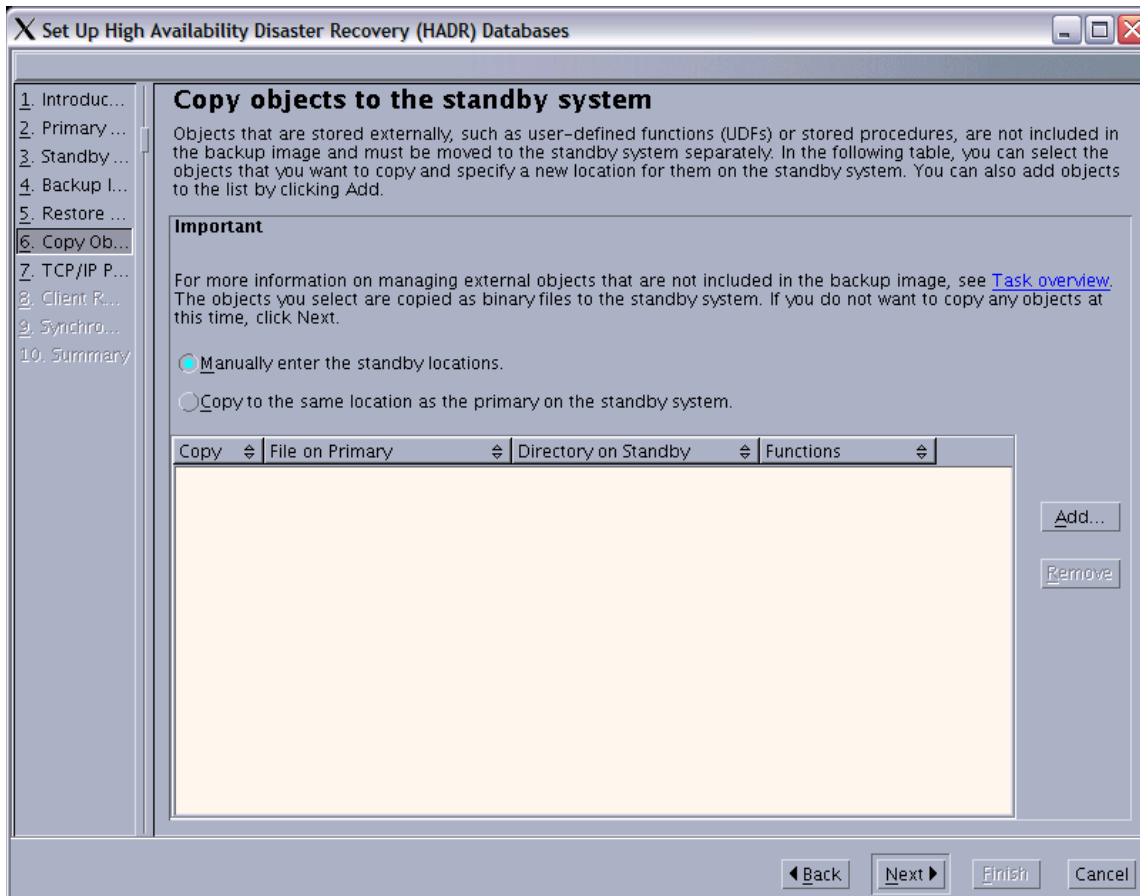


Figure 3-8 Copy “special” objects across to the standby system

9. *Specify TCP/IP communication parameters:* Figure 3-9 shows the settings required for the HADR-specific host name and service/port number, for the primary and standby databases. Enter unique values as required and click **Next**.

**Tip:** The wizard automatically inserts values for the HADR services in the `/etc/services` file. If you run the wizard more than once or have these services already registered, the service port numbers in this dialog box are incremented (for example, 55003 and 55004, if we ran this same wizard again after using 55001 and 55002 the first time). Ensure that you manually change the values back as necessary on subsequent runs.

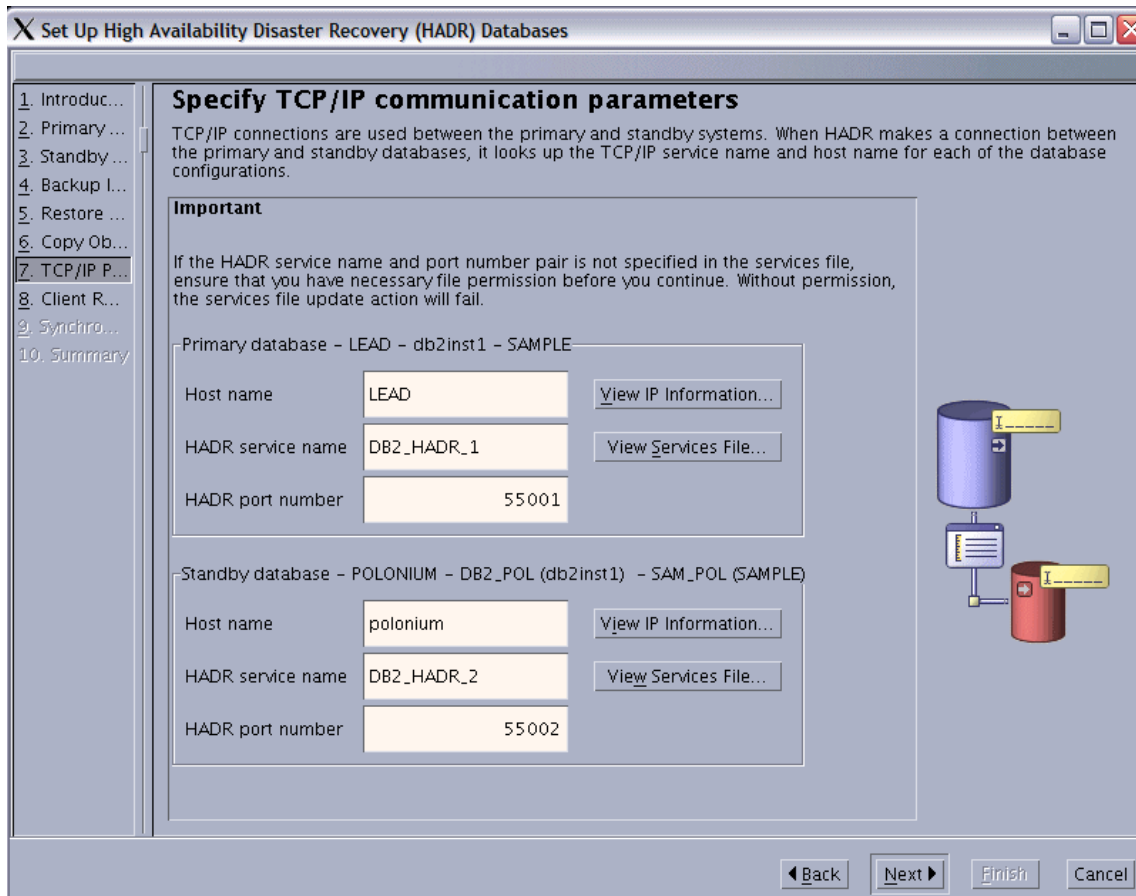


Figure 3-9 Specify host addr and port numbers for both HADR communication channels

10. *Configure databases for automatic client reroute*: Figure 3-10 shows where we set up the IP address redirect facility: where each client connection to a DB2 server stores the primary IP address and port number of the primary server, and also of the standby server. When you switch roles, or force a takeover, the other IP address is then used by the client; all the client sees is an SQL message to inform that the connection has changed and the transaction or database connection should be retried. See Chapter 5, “Automatic client reroute” on page 121 for more details on ACR.

The port number refers to the main port number for the DB2 instance, not to the HADR port. Just match the port number with the correct server name/IP address, and remember that the “alternate” for the primary is the standby server, and vice versa. Click **Next**.

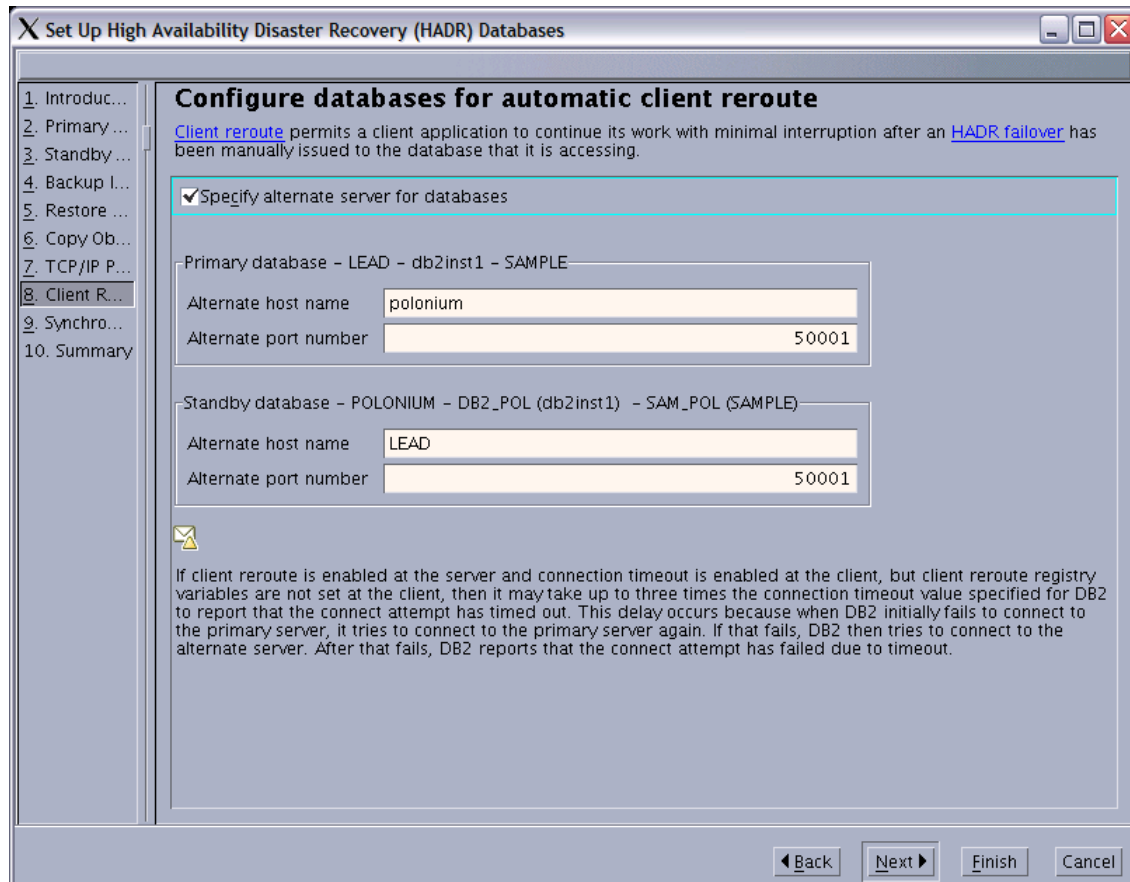


Figure 3-10 Configuring Automatic Client Reroute server and port information

11. *Specify synchronization mode for Peer state log writing:* Figure 3-11 lets us choose whether we want Synchronous, Near-Synchronous, or Asynchronous mode for HADR. We choose Synchronous for our example. See 2.4, “HADR architecture” on page 39 for more information on the three available synchronization modes. Specify the timeout value. For details of the best practice on choosing the synchronization mode and recommendations on how to speed up takeover time, see Chapter 6, “HADR best practices” on page 143.

Click **Next**.

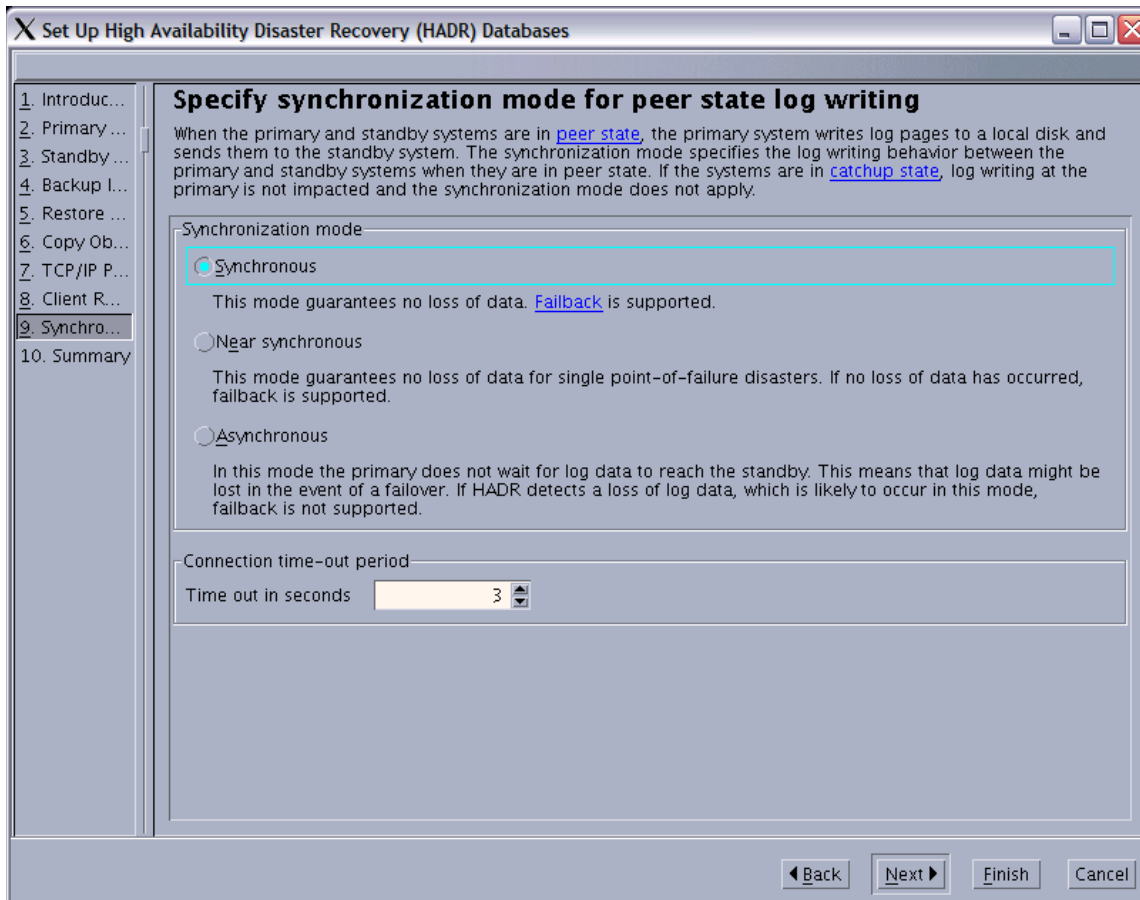


Figure 3-11 Choose an HADR syncmode: SYNC, NEARSYNC, or ASYNC

12. Review the actions that occur when you click *Finish* (Figure 3-12):

This dialog box shows an overview of the settings chosen for this HADR configuration. Click **Show Command...** to get command line statements in order to learn what goes on behind the scenes in the wizard. These commands are worth saving in a text file, certainly for subsequent configurations.

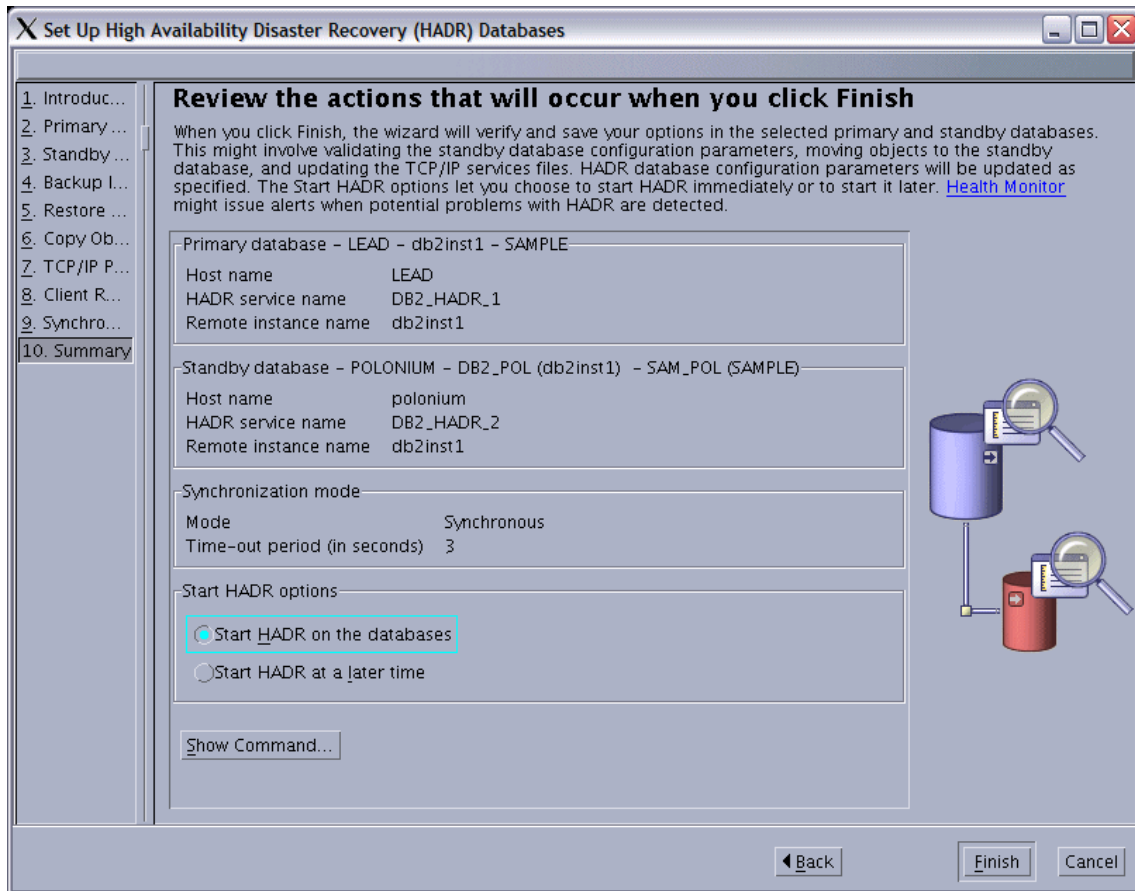


Figure 3-12 Reviewing the proposed configuration before commencing final HADR scripted steps

You can choose to either have the wizard start HADR, or issue that command yourself afterward through the Manage HADR wizard or on the command line.

Click **Finish**.

**Note:** The Show Command only takes into consideration the work to be done after configuring linear/archival logging, so prerequisite updates to the database configuration parameters LOGARCHMETH1/2 and LOGINDEXBUILD are not listed.

13. *Execution of Steps*: Figure 3-13 shows a dynamic summary of those commands being generated and run by the automated script. If any issues are encountered, the wizard should let you know what must be corrected, and you should be able to click **Finish** on the previous dialog box in order to retry the process.

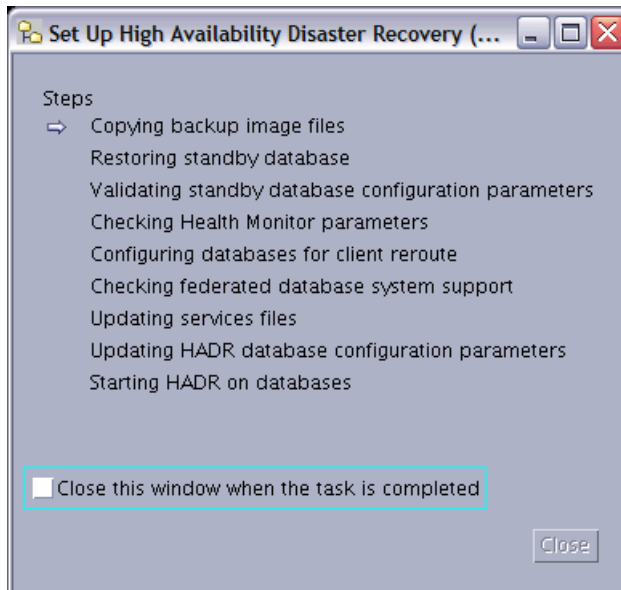


Figure 3-13 Commencing the steps to process a scripted HADR setup

Figure 3-14 shows the completed steps, which is what you should expect. HADR setup is now complete. Click **Close**.



Figure 3-14 The completed steps for a scripted HADR setup

14. *Verify HADR setup*: To confirm that HADR is up and running, you can check with the DB2 Control Center.

First, select **View** → **Refresh**, otherwise it displays the remote database as “unknown”. Next, right-click either of the HADR databases and select **High Availability Disaster Recovery** → **Manage HADR**. The result should be similar to Figure 3-15.



If you left the choice to start HADR as blank in step 11 (Figure 3-12 on page 68), you can click **Start HADR...** here, which should then show your two databases in a Peer state, with the appropriate synchronization mode, the connected state, and the log information on each. You can get similar information using the GET SNAPSHOT command or db2pd.

**Note:** When using DB2 Version 8, we recommend at least FixPak 14 to correct SYNCMODE value discrepancies in the Manage HADR dialog.



Figure 3-15 In Control Center: Manage HADR

### 3.2.3 Command line setup

DB2 provides both commands and GUI setup wizard for HARD. In this section, we demonstrate how to set up HADR using commands. In our Lab environment, we set up the LEAD server as the primary server, and POLONIUM server as the standby with the database SAMPLE on both the servers.

To set up HADR for the command line environment, complete the following steps:

1. Set the required database configuration parameters.

If archive logging is not already turned on, then update the LOGRETAIN parameter using the following command:

```
db2 update database configuration for sample using LOGRETAIN
recovery
```

Also, set the LOGINDEXBUILD parameter, so that index creation, re-creation, or reorganization operations are logged, using the following command:

```
db2 update database configuration for sample using LOGINDEXBUILD ON
```

2. Back up your database on the primary and move a copy to the standby server using the following command:

```
db2 backup database sample to /usr/db2/backup
```

3. Restore the database on your standby system.

In our example, we restore to the SAMPLE database on the POLONIUM server using the following command:

```
db2 restore database sample from /usr/db2/backup taken at
20061011141321 replace history file
```

4. Configure databases for ACR. This step is optional but highly recommended. To configure ACR, update the ALTERNATE SERVER database configuration parameter on both the primary and standby servers with the following steps:

- a. On the primary server, LEAD in our test case, set the standby server as the alternate server using the following command:

```
db2 update alternate server for database sample using hostname
polonium port 50001
```

- b. On the standby server, POLONIUM in our test case, set the primary server as the alternate server using the following command:

```
db2 update alternate server for database sample using hostname
lead port 50001
```

5. Update the following fields in the services file on the primary system LEAD for HADR communication:

- Service name: DB2\_HADR\_1
- Port number: 55001
- Service name: DB2\_HADR\_2
- Port number: 55002

6. Update the following fields in the services file on the standby system POLONIUM for HADR communication:
  - Service name: DB2\_HADR\_1
  - Port number: 55001
  - Service name: DB2\_HADR\_2
  - Port number: 55002
  
7. Update the HADR database configuration parameters on the primary database with the following commands. In our test case, it is LEAD.
 

```
db2 update db cfg for sample using HADR_LOCAL_HOST LEAD
db2 update db cfg for sample using HADR_LOCAL_SVC DB2_HADR_1
db2 update db cfg for sample using HADR_REMOTE_HOST POLONIUM
db2 update db cfg for sample using HADR_REMOTE_SVC DB2_HADR_2
db2 update db cfg for sample using HADR_REMOTE_INST DB2INST1
db2 update db cfg for sample using HADR_SYNCMODE SYNC
db2 update db cfg for sample using HADR_TIMEOUT 3
db2 update db cfg for sample using HADR_PEER_WINDOW 120
db2 connect to sample
db2 quiesce database immediate force connections
db2 unquiesce database
db2 connect reset
```
  
8. Update the HADR database configuration parameters on the standby database, POLONIUM in our test case, using the following commands:
 

```
db2 update db cfg for sample using HADR_LOCAL_HOST POLONIUM
db2 update db cfg for sample using HADR_LOCAL_SVC DB2_HADR_2
db2 update db cfg for sample using HADR_REMOTE_HOST LEAD
db2 update db cfg for sample using HADR_REMOTE_SVC DB2_HADR_1
db2 update db cfg for sample using HADR_REMOTE_INST db2inst1
db2 update db cfg for sample using HADR_SYNCMODE SYNC
db2 update db cfg for sample using HADR_TIMEOUT 3
db2 update db cfg for sample using HADR_PEER_WINDOW 120
```
  
9. Start HADR on the standby database on POLONIUM in our test case, using the following commands:
 

```
db2 deactivate database sample
db2 start hadr on database sample as standby
```
  
10. Start HADR on the primary database on LEAD in our test case, using the following commands:
 

```
db2 deactivate database sample
db2 start hadr on database sample as primary
```

## 3.3 Basic operation

This section give you basic details on the start, stop, and takeover operations preformed for HADR. We give examples for both the command line environment and GUI environment.

### 3.3.1 Starting up and shutting down

This section covers the startup and shutdown procedures for HADR on a GUI and command line environment.

#### Startup

Before starting HADR, you should ensure that your database manager (instance) on both the primary and standby databases is started. Use the **db2start** command to start the instance. The instance can be started in any order, primary or standby.

When starting HADR, we recommend that you start the standby database before the primary database. The reason for starting the standby first is that the primary HADR startup, without the BY FORCE option, requires the standby to be active within the HADR\_TIMEOUT period or startup fails to prevent a split-brain scenario.

#### *Using the startup command*

This is the syntax for the startup command:

```
START HADR ON DATABASE database-alias [USER username [USING password]]  
AS {PRIMARY [BY FORCE] | STANDBY}
```

When starting the primary, the BY FORCE option specifies that the HADR primary database does not wait for the standby database to connect to it. After a start BY FORCE, the primary database still accepts valid connections from the standby database whenever the standby later becomes available.

For example, to start HADR on the primary, issue the following command:

```
db2 start hadr on database sample as primary
```

It is important to note which database you are on when you start HADR.

You can find the details for the START HADR command in the *DB2 9.1 Command Reference*, SC10-4226.

## Starting from the Control Center

HADR can also be started from the HADR wizard in the Control Center:

1. From the Control Center, expand the object tree down to the database object, right-click the database name you plan to start HADR on. Select **High Availability Disaster Recovery** → **Manage** as shown in Figure 3-16.

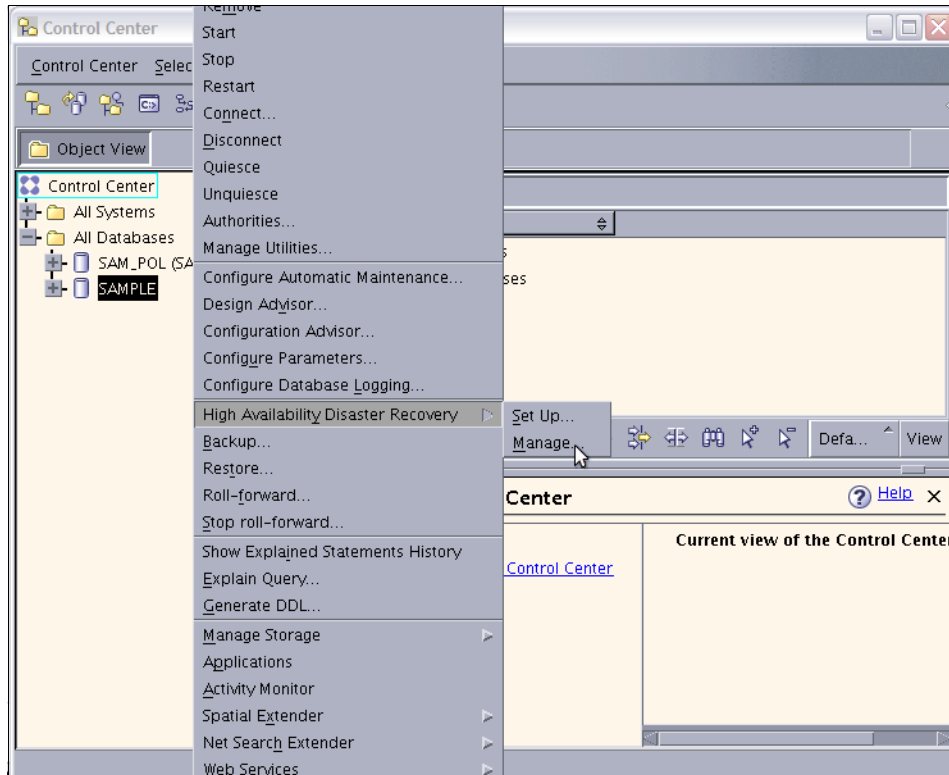


Figure 3-16 Manage HADR

- From the Manage High Availability Disaster Recovery (HADR) window, click **Start HADR** as shown in Figure 3-17.

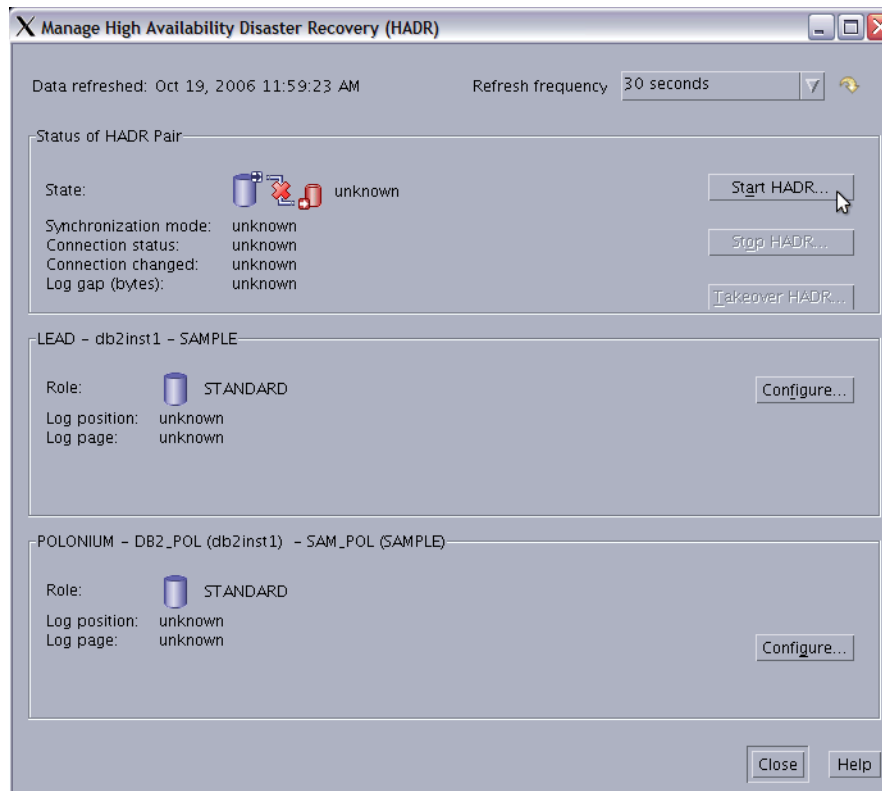


Figure 3-17 Start HADR from Manage screen

3. From the Start HADR window shown in Figure 3-18, you can choose to start the primary or the standby only, or both. Click **OK**.

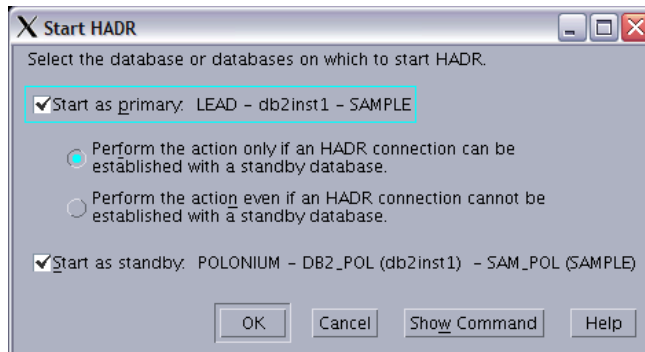


Figure 3-18 Start HADR

**Note:** When starting the primary using the choice, Perform the action even if an HADR connection cannot be established with a standby database, this issues the BY FORCE option in the START HADR command.

## Shutdown

Although the STOP HADR command can be used to stop HADR on the primary or the standby, or both, it should be used with caution. If you want to stop the specified database but still want it to maintain its role as either an HADR primary or a standby database, do not issue the STOP HADR command. If you issue the STOP HADR command, the database becomes a standard database and might require re-initialization in order to resume operations as an HADR database. Instead, issue the DEACTIVATE DATABASE command.

If you only want to shut down the HADR operation, this is the recommended way of shutting down the HADR pair:

1. Deactivate the primary database.
2. Stop DB2 on the primary database.
3. Deactivate the standby database.
4. Stop DB2 on the standby database

### ***Using the STOP HADR command***

You can stop HADR from the command line or the Control Center. When you want to bring your database to a standalone database, you can use the STOP HADR command.

The shutdown command has the following syntax:

```
STOP HADR ON DATABASE database-alias [USER username [USING password]]
```

For example:

```
db2 stop hadr on database sample
```

You can find the details for the STOP HADR command in the *DB2 9.1 Command Reference*, SC10-4226.

### **Stopping HADR from the Control Center**

You can stop HADR with the HADR wizard by performing the following steps:

1. From the Control Center, expand the object tree down to the database object. Right-click the database name you plan to stop HADR on. Select **High Availability Disaster Recovery** → **Manage** (refer to Figure 3-16 on page 75).
2. From the Manage High Availability Disaster Recovery (HADR) window, click **Stop HADR** as shown in Figure 3-19.

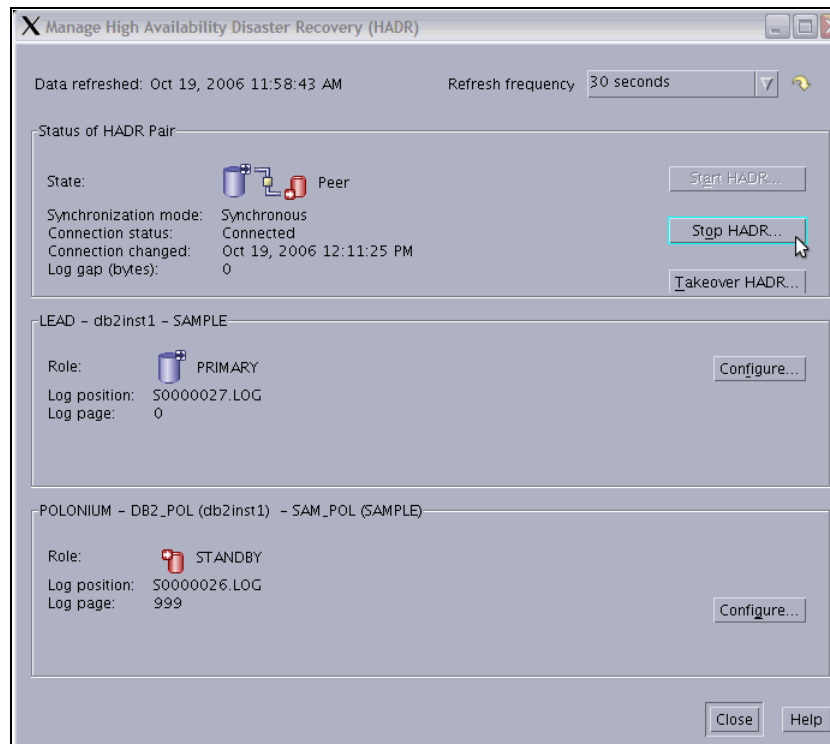


Figure 3-19 Manage HADR stop



- From the Stop HADR window shown in Figure 3-20, you can choose to stop just the primary, or both the primary and the standby. If one of the databases has HADR stopped, the Stop HADR window only allows you to stop the database where HADR is started. Click **OK**.

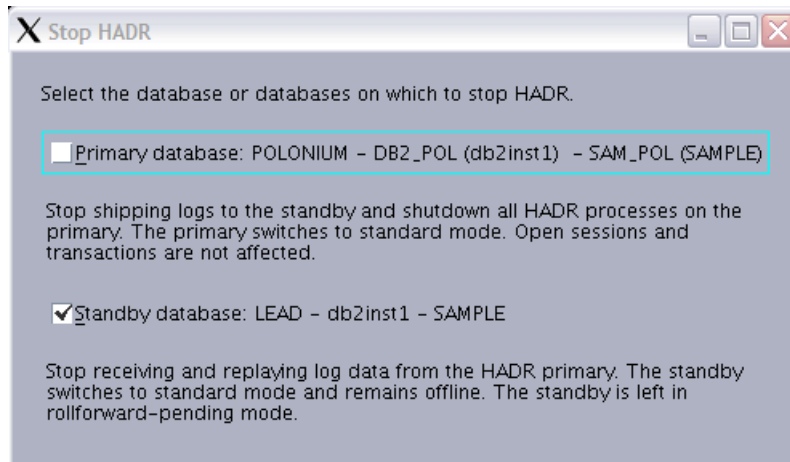


Figure 3-20 Stop HADR

### 3.3.2 Planned takeover

Planned takeover is also referred to as switching roles or role exchange. You issue the TAKEOVER command when you want to switch roles of the databases. For example, during an upgrade you switch roles, making the standby the new primary. Remember to re-route your clients either manually or by using the ACR after you have issued the TAKEOVER command.

Switching roles is only done from the standby when the databases are in the Peer state. The TAKEOVER command fails if the databases are in any other state.

The takeover procedure is simply issuing the TAKEOVER HADR command.

For example, issue the takeover command on the standby database using the following command:

```
db2 takeover hadr on database sample
```

The details for the TAKEOVER HADR command are found in the *DB2 9.1 Command Reference*, SC10-4226.

After issuing the TAKEOVER HADR command from the standby, the following steps are carried out in the background:

1. Standby tells the primary that it is taking over.
2. Primary forces off all client connections and refuses new connections.
3. Primary rolls back any open transactions and ships the remaining log, up to end of the log, to standby.
4. Standby replays received log, up to end of the log.
5. Primary becomes the new standby.
6. Standby becomes the new primary.

### Using the GUI

In our example, we start with SAMPLE database on POLONIUM as the primary database. Perform the following steps:

1. From the Control Center expand the object tree down to the database object, right-click the database name you plan to takeover HADR on. Select **High Availability Disaster Recovery** → **Manage**.

- From the HADR window on the standby (POLONIUM), click **Takeover HADR** as shown in Figure 3-21.

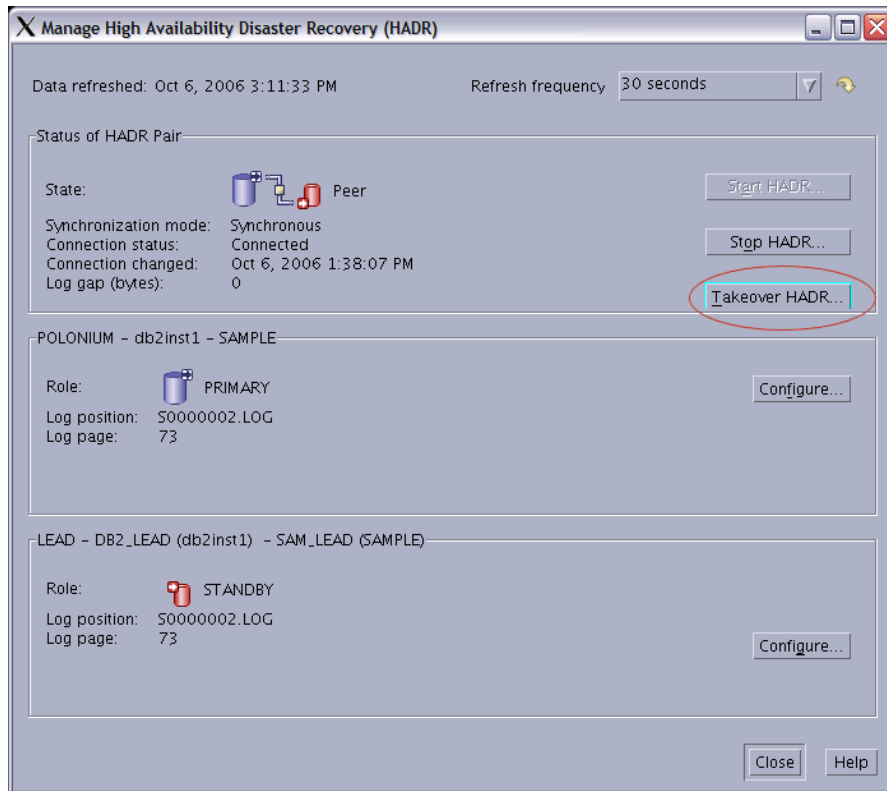


Figure 3-21 Manage HADR Takeover

3. From the Takeover HADR window, select **Switch roles** as shown in Figure 3-22 and click **OK**.

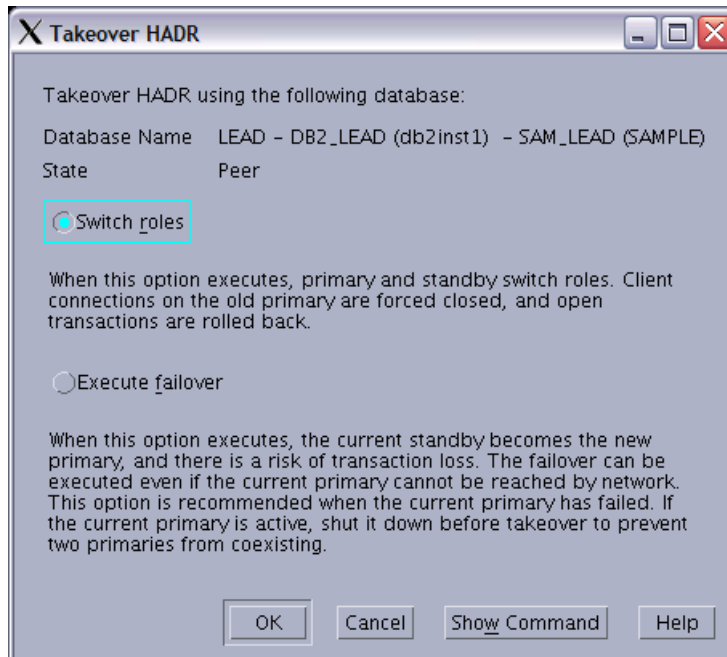


Figure 3-22 Takeover HADR

Clicking **Show Command** provides you the command DB2 used for this operation. Figure 3-23 illustrates the output.

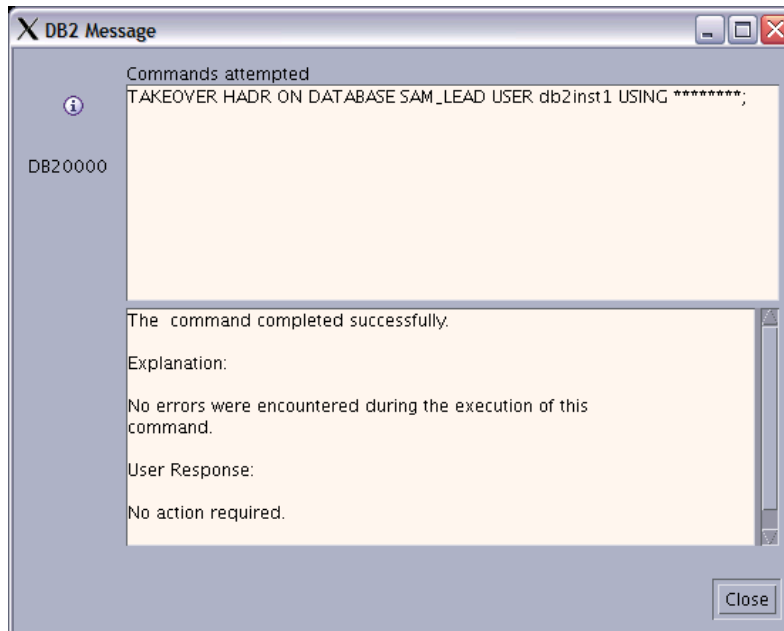


Figure 3-23 Takeover HADR command

Now the primary and the standby roles are swapped. Issue the same command again to swap back the roles. This can be done as often as necessary, so long as the databases are in the Peer state.

### 3.3.3 Takeover by force

A *takeover by force* is also referred to as *failover*, which is issued from the standby database with the TAKEOVER command BY FORCE option included. You should use the takeover by force only if your primary database is not functional. The BY FORCE option tells the standby to become the new primary without coordinating with the original primary as it does with a planned takeover. When the PEER WINDOW ONLY sub-option is specified, there is not any committed transaction loss if the command succeeds and the primary database is brought down before the end of the peer window period.

The procedure for failover includes the following steps:

1. Make sure that the primary is down to minimize the chances of data loss. If a takeover BY FORCE is issued and the primary is not down, this can result in both the databases becoming primary databases. This is referred to as *split brain*.
2. Issue the TAKEOVER HADR command with the BY FORCE PEER WINDOW ONLY option on the standby database. The following is an example for the sample database:

```
db2 takeover hadr on database sample by force
```

After issuing the TAKEOVER HADR command with the BY FORCE option from the standby, the following steps are carried out in the background:

1. The standby sends a notice asking the primary to shut down.
2. The standby does *not* wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down.
3. The standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

Data loss is possible when a takeover by force is issued. Your chance of data loss depends on your configuration and circumstances. The following list shows the general settings and the result of a failure on the primary.

- ▶ If the primary database is in the Peer state when it fails:
  - With syncmode set to SYNC, the standby does not lose any transactions that were reported committed to the application before the primary failed.
  - With syncmode set to NEARSYNC, the standby only loses transactions committed by the primary if the primary and standby databases fail at the same time. This is a highly unlikely circumstance.
  - With syncmode set to ASYNC, the standby database can lose transactions committed on the primary if the standby did not receive all of the log records for those particular transactions before the takeover operation was performed. As in the NEARSYNC mode, if the primary and the standby fail at the same time, transactions can be lost.
- ▶ If the primary is in the remote catchup pending state or any other non-Peer state when it fails:
  - For all the three syncmodes (SYNC, NEARSYNC, ASYNC), transactions that have not been received and processed by the standby database is lost.

When you issue the `TAKEOVER BY FORCE PEER WINDOW ONLY` command, and it succeeds, then there is not any transaction information on the primary database that was not already copied to the standby database. This ensures greater degree of data consistency.

For more details on the ways to prevent data loss in a forced takeover, see Chapter 7 in *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228.

Following a takeover by force because of a failure at the primary database, after the old primary is recovered and you bring it back online, you can reintegrate the old primary as the standby database.

To reintegrate the old primary, perform the following steps:

1. Recover the failed primary, and bring it back online.
2. Restart the failed primary as the new standby using the `START HADR` command, for example:

```
db2 start hadr on database sample as standby
```

If the two databases have incompatible log streams, for example, because of logs not being received from the standby before takeover, then the reintegration of the old primary to the new standby fails and you have to restore a backup of the current primary to your failed primary to start it as the new standby. Refer to 3.4.5, “Re-establishing HADR after failure” on page 92 for more details on reintegration.

### Example of takeover by force in GUI

To perform a takeover by force in the GUI, complete the following steps:

1. From the Control Center, expand the object tree down to the database object, right-click the database name you plan to takeover HADR on. Select **High Availability Disaster Recovery** → **Manage** (refer to Figure 3-16 on page 75).
2. From the Manage High Availability Disaster Recovery (HADR) window on the standby (POLONIUM in our test case), click **Takeover HADR** as shown in Figure 3-21 on page 81.

- From the Takeover HADR window, select **Execute failover** as shown in Figure 3-24 and click **OK**.

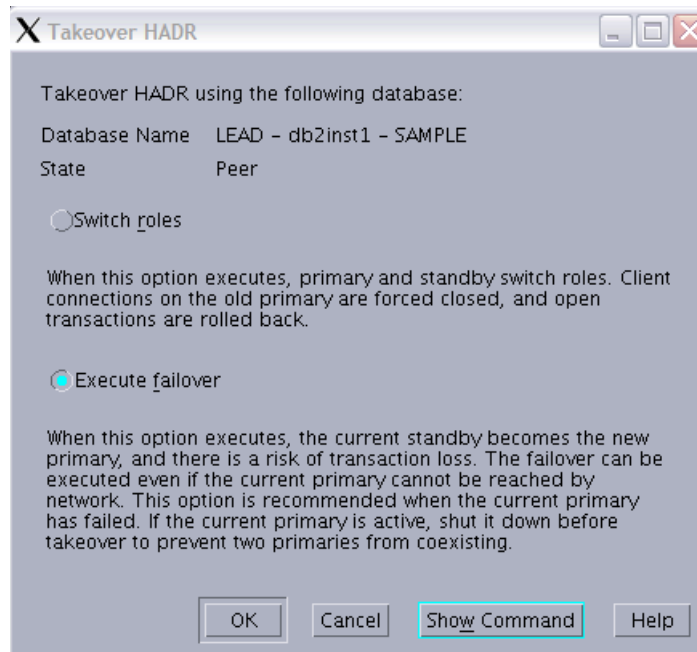


Figure 3-24 Takeover HADR by force

## 3.4 Troubleshooting

Here we examine some of the possible issues you might encounter while setting up and running HADR, and while attempting to resume the HADR operation after a server failure, or other unanticipated interruption to DB2 HADR communication. In most cases, the solutions and workarounds are straightforward, but not often immediately obvious from any error messages you might or might not be getting at the time. These examples are from first-hand experience, and the workarounds presented might not be the only way to resolve the situation.

An additional source of material to review in case you run into trouble when setting up, are the platform-specific prerequisites and maintenance dependencies which are dynamically updated online at the following Web site:

<http://www.ibm.com/software/data/db2/9/sysreqs.html>

[http://www.ibm.com/software/data/db2/9/sysreqs\\_v8.html](http://www.ibm.com/software/data/db2/9/sysreqs_v8.html)



### 3.4.1 During setup

Here is a list of common issues and possible fixes encountered during setup:

► Symptom:

On AIX/UNIX/Linux, error messages are indicating that graphical tools such as the Control Center cannot be started due to missing Java libraries.

– Fix:

For 64-bit DB2 V8, AIX system support might have to install 64-bit Java libraries, and then you set the admin cfg and dbm cfg value of `JDK_PATH`, and admin cfg value of `JDK_64_PATH` appropriately, and restart the DB2 instance(s) and DB2 Admin Server.

DB2 9 comes with built-in Java libraries, so this problem should not occur.

► Symptom:

On Windows, various error messages are claiming that the user ID does not have the authority to execute the given command, even though the user ID is both the DB2 instance ID with implicit `SYSADM` authority, and a Windows Administrator.

– Fix:

As a prerequisite, ensure that if you are in a permanently connected domain, have the `DB2ADMNS` group with the DB2 instance user ID created at least on the domain and optionally replicated onto the local servers as a security policy. Ensure that you are logged on to Windows as the same user ID type as DB2 was installed with. In an Active Directory® configured server, most likely this is the domain user ID rather than the local user ID. Ensure that the user ID you log on with is part of the Administrator's group. You can also encounter issues with the DB2 instance user ID's implicit authority if you change the `SYSADM_GROUP` db cfg parameter from the default blank value, even if the DB2 instance ID is connected to the `SYSADM_GROUP`.

If you are running DB2 on a server that is configured in a domain, but is not permanently connected to that domain and does not have a security policy periodically overwriting the local registry, then it might be better to install everything using the local user ID. A final consideration is to check the value of the DB2 registry variable `DB2_GRP_LOOKUP`, which is specific to the Windows environment. Users who log on as the domain user ID have their group connections in Windows checked against the domain groups by default, rather than local groups. Setting the `DB2_GRP_LOOKUP` parameter to `LOCAL` overrides this behavior.

► Symptom:

You cannot see the remote server or DB2 objects on the remote/standby end when attempting to discover DB2 instances and databases.

– Fix:

Start DB2 Admin Server on the remote end, and confirm firewall is open for port 523, and/or manually catalog the administrative TCP/IP node, the instance TCP/IP node, and the remote database on the command line. See Example 3-4. If you wish to manage or monitor HADR via the Control Center, you require the Admin Server to be working at both ends for connectivity.

*Example 3-4 Catalog TCP/IP node and database*

---

```
db2 catalog admin tcpip node polonium remote x.x.x.x remote_instance
db2inst1 system polonium
db2 catalog tcpip node db2_pol remote x.x.x.x server 50001
remote_instance db2inst1 system polonium
db2 catalog database sample as sam_pol at node db2_pol
```

---

If you execute the commands to catalog nodes and database manually in the middle of the Setup HADR dialog boxes, you need to cancel out to the Control Center main dialog box, select **View** → **Refresh**, and restart, in order to see the remote objects in the dialog boxes.

### 3.4.2 After setup or during normal execution

The following common issues and possible fixes are encountered after setup and when HADR is operating:

► Symptom:

Cannot see the remote server or DB2 objects on the remote/standby end in the Control Center Manage HADR dialog box. The values are displayed as “unknown”, even when DB2 on the remote server is definitely up and running, and HADR status displays as connected and in the Peer state in output from **db2pd** or **get snapshot for db** commands.

– Fix:

- i. The first thing to check is that DB2 Admin Server is started on the remote server.
- ii. Next, confirm that the ADMIN TCPIP and TCPIP NODEs and remote database are cataloged correctly.

- iii. Ensure that the server name or IP address as specified in the cataloged node names is actually registered in both the local and the remote `/etc/hosts` files, and that they match the host name or IP address as registered in the remote server's `<DB2 Instance home>/sqlib/db2nodes.cfg`. By experience we have found that servers on VM or similar architectures can have multiple IP addresses defined, only some of which are visible through firewalls, and which might not match the host name a remote administrative client should be able to see.

► Symptom:

SQL30081N Communication error has been detected when attempting to connect to the database on the remote server, even though previously there was no issue, and both the relevant DB2 instance and DB2 Admin Server are definitely up and running on the remote server.

– Fix:

This is an extremely common issue on a dynamic network environment, and could be caused by any number of reasons, including firewall issues, IP address or port number mismatches or conflicts, incorrect `/etc/hosts` or `/etc/hosts.allow/deny` content, host name or DNS issues. The first thing to do is to try whatever the Messages Reference recommends for any protocol specific error codes (refer to the *DB2 9.1 Message Reference Volume 2*, SC10-4239, Chapter 2). After exhausting that, you might try falling back from usage of DNS addresses and service names, and temporarily use the direct IP address and port numbers instead, which you know to be correct and open in the firewall. If this issue persists, ask whomsoever supports the network architecture to check that the firewall ports are open and whatever else might be causing the described communication error symptoms, providing them with all the protocol specific error message information.

► Symptom:

Synchronization mode on the Manage HADR dialog box does not match the output from `db2pd`, `get db cfg`, or `get snapshot for db`. NEARSYNC appears correctly, but SYNC and ASYNC appear to be swapped.

– Fix:

This is a known cosmetic error isolated to the Manage HADR dialog, which has now been corrected. We recommend using DB2 V8 FixPak 14 in order to display the correct SYNCMODE value.

► Symptom:

After changing `db cfg` parameters on both servers, performing deactivation and reactivation on the standby, HADR still does not reconnect, so that roles can be switched for the second database deactivation/reactivation:

- Fix:

If you are changing certain parameters, such as any of the HADR\_ prefixed db cfg parameters, the currently active values must match on both sides in order for HADR connectivity to work. You require downtime on the primary after changing the db cfg parameter on both the servers to establish the new value as current. Plan ahead for an outage window to achieve this, otherwise you risk rendering your standby inoperative and disconnected until you change the db cfg value back again to match the current value on the primary.

### 3.4.3 After an HADR disconnect or server failure

Here is an issue and possible fixes encountered after HADR is disconnected or a server failure.

- ▶ Symptom:

As an example, let us say an operating system fix is carried out, and the server gets recycled without so much as a courtesy e-mail to warn DB2 support about it. As a result, HADR becomes disconnected, and attempts to restart it continually fail.

- Fix:

In this case, most likely the cause is mismatching or lost log records on the standby, such that it cannot reintegrate successfully with the primary. You can find this out by just looking at the output in the db2diag.log file for the times you attempt to start HADR on either the primary or the standby. The error messages for this are clear. Subsequent corrective action is straightforward.

You have no choice but to re-establish HADR by backing up the primary, transferring it to the standby, and restoring it there. Instructions for this are fully set out in 3.4.5, “Re-establishing HADR after failure” on page 92.

### 3.4.4 Considerations while running HADR

When the HADR is configured and running, consider these situations:

- ▶ If you schedule backup and housekeeping tasks through the DB2 tools database, you should consider managing it through HADR in parallel with your application database. In the case where an extended outage of the primary server carries over through a critical batch window, you want it to be working and running the same tasks just as on the original primary server.
- ▶ The db cfg parameter AUTORESTART should be turned off if you do not want the standby database to attempt to roll forward through logs while HADR

is stopped and the DB2 instance gets recycled while that database is still in STANDARD mode.

- ▶ If you use ACR rather than a clustering solution to control IP addressing and takeover for remote clients, **db2iauto** should be turned off for the DB2 instances which control HADR databases. To explain using an example of server A and server B, this is so that DB2 on a broken primary server A does not automatically restart when server A is subsequently fixed and started up, causing a split-brain scenario.

In the context of HADR, *split brain* can be summarized as where remote clients cannot distinguish one server from the other when connecting to their database, opening the possibility of changes being made to the wrong database, and then being unable to be isolated and transferred to the correct database.

Fortunately, DB2 HADR is able to protect itself from split-brain scenarios where an HADR primary database has been deactivated while it was in HADR primary mode. If a client attempts to connect to such a database, it goes through an attempt to activate the database in HADR primary mode, and connect to an HADR standby. After the HADR\_TIMEOUT interval, DB2 returns an SQL1768N with reason code 7:

The primary database failed to establish a connection to its standby database within the HADR timeout interval.

This is also why it is easier from an Administration point of view to start the HADR standby database first, then the primary, as there is no pressure to get the other database started before the HADR\_TIMEOUT interval lapses and the database is left inactive again.

Split brain in the context of HADR can still occur in other ways:

- Where the broken HADR primary server was never actually deactivated (for example, where the failure was with the network rather than with the server), and after a **db2 takeover hadr on database ... by force** command is issued by the standby server B, the broken primary server A unexpectedly comes back online.

A proper clustering solution should be able to cater to this problem by use of resource group takeover actions, where a primary server that finds itself isolated knows to relinquish ownership of resources and issue shutdown scripts to critical resources on that server, but there is no easy fix in a non-clustering scenario. This can also happen irrespective of any temporary failure — for example, when the forced takeover command is issued on server B but the HADR primary on server A was never stopped.

- Where the broken HADR PRIMARY on server A has a **db2 stop hadr on database** command issued on it, such that it is now in STANDARD mode, and clients can once again connect to it instead of the correct HADR

primary database running on server B. As with the previous scenario, this should not be possible in a proper clustering solution that manages IP addressing. Remote clients in this case can only connect to a single server that has the role of the active node.

### 3.4.5 Re-establishing HADR after failure

Also referred to as *reintegration* of HADR, this process is very similar to the initial setup of HADR, but without all the configuration of db cfg parameters.

There are two ways to successfully get HADR working again in a paired primary and standby relationship; both should require no downtime, and the first is an extremely rapid and low complexity solution.

For familiarity, we are using the example of HADR database SAMPLE on both Server A as primary and Server B as standby, where Server A suffers a failure, followed by a **takeover hadr by force** command on server B, such that server B becomes the new primary.

The first way only works if the primary and the standby were in Peer state at the time of server A's failure. In this scenario, the DB2 instance is eventually restarted on server A after correcting whatever failure occurred. DB2 HADR protects itself from split brain by not allowing activation of the database on server A, returning SQL1768N with RC 7 if any database activation or remote client connection attempts are made at this time.

The next thing to do is to issue the command **db2 start hadr on database sample as standby** on Server A. Note that database SAMPLE also cannot have had a **db2 stop hadr on database sample** command run beforehand, as it then contains log records that server B does not know about, making it effectively unusable as a standby for server B. Attempts to restart HADR on it as standby result in SQL1767N RC 1. Attempting to start HADR as standby on server A's SAMPLE database naturally fail for the same reason if the databases were not in Peer state at the time of server A's failure.

In an ideal situation, the Peer state should be continually tracked on the HADR primary server. If any failure occurs while not in Peer state, then consideration should be made before issuing any **db2 hadr takeover database sample by force** command on the standby server B. In this situation, every effort should be made to correct issues on Server A and have that database restarted rather than issue a takeover by force command on Server B's STANDBY database, as Peer state is the only guarantee you have that there has been no lost data from committed transactions.

The second way works regardless of whether the HADR pair was in Peer state at the time of a server failure. This method is relatively less complex, and saves time compared to the initial setup of HADR, because the db cfg parameters are stored separately from the physical data containers, in a non-user-readable file called SQLDBCONF. This file is not replaced by a restore command unless that database has been dropped prior to the restore.

At the most, if you need to drop your standby database before executing the restore from primary backup, you would need to change the following db cfg parameters:

- ▶ HADR\_LOCAL\_HOST = <hostname>
- ▶ HADR\_LOCAL\_SVC = <this server's HADR port# >
- ▶ HADR\_REMOTE\_HOST = <the remote server hostname
- ▶ HADR\_REMOTE\_SVC = <the remote server's HADR port#>
- ▶ HADR\_REMOTE\_INST = <the remote server's DB2 Instance name>

Normally, you would never need to drop your standby database prior to the restore. However, DB2 does not let you restore over the top of a database which is currently in an HADR standby or primary state, and in some cases, DB2 does not let you stop HADR and put that database into a standard mode; this can happen if HADR was not stopped on a database before the DB2 instance is migrated from version 8 to version 9 for example.

Apart from not having to change your db cfg parameters after a restore, one reason you would not want to drop the database before the restore would be if you have a database with large containers, irrespective of the amount of data contained within them. DB2 needs time to pre-format at the time of database creation. Large containers take more time. The time needed depends on the storage and system capacity. Restoring the database without dropping the database beforehand would save the pre-format time.

If you are concerned about scheduling a downtime window for re-establishing HADR, you can rest assured that you should not need one: at most, you might experience some read-only impact for a short duration while the online backup takes place on the primary database. As *DB2 9.1 Command Reference*, SC10-4226, Chapter 3 states:

“During an online backup, DB2 obtains IN (Intent None) locks on all tables existing in SMS table spaces as they are processed and S (Share) locks on LOB data in SMS table spaces. “

## Command line steps

Re-establishing (reintegrating) the standby with the primary can be achieved through a few command line steps issued by the DB2 instance user ID:

1. Prepare a database backup to restore the broken standby with:

Backup your primary database, online or offline, it does not matter. You might wish to compress the backup to save time when transferring it to the remote (standby) server, or use a TSM target which the remote server has been authorized to read from. In our example on Linux, we make a compressed backup of the SAMPLE database on the local server (LEAD) to local disk, then use **scp** to transfer the backup file to the remote server (POLONIUM):

```
db2 backup database sample online to /usr/db2/backup compress
```

2. Transfer your primary database backup file to the remote server:

For our example, we need to find the name of the backup file; the destination of the backup step was `/usr/db2/backup`. In Windows, up to and including DB2 V8, the file name of the backup files consists of a hierarchy of subdirectories inside the destination subdirectory. The file name to be copied is only the final portion of the `hhmmss` portion plus a three digit extension indicating the backup sequence number, usually `.001`, unless you perform concurrent backups. As of DB2 9, the backup file name on Windows matches Linux/UNIX, a long file name in a single subdirectory.

In Example 3-5, Linux server, `ls -l` of `/usr/db2/backup` shows:

### Example 3-5 List database backup files

---

```
db2inst1@lead:/usr/db2/backup> ls -l
total 53380
-rw-r----- 1 db2inst1 db2grp1 12603392 2006-10-05 18:26 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061005182651.001
-rw-r----- 1 db2inst1 db2grp1 12079104 2006-10-11 13:46 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061011134630.001
-rw-r----- 1 db2inst1 db2grp1 12079104 2006-10-11 14:13 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061011141321.001
-rw-r----- 1 db2inst1 db2grp1 17846272 2006-10-18 15:11 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001
```

---

The backup file name that we want contains a matching timestamp with the output of the above example backup command as follows:

Backup successful. The timestamp for this backup image is : 20061018151120

The matching file for us is:

```
SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001
```

The timestamp (20061018151120, in our example), is also used in the restore command later, so note it down.

Now we actually issue the transfer command, whether that be **scp** as shown in our example, or **sftp** or some other method:

```
scp SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001
db2inst1@polonium:/usr/db2/backup
```



3. We log on to the remote server (POLONIUM in our example) and issue the restore database command. Note that HADR is not able to re-establish a connection if the database on the standby end is rolled forward after the restore:

```
db2 restore db sample from /usr/db2/backup taken at 20061018151120
```

As you can see, we use the timestamp in the restore command. Respond “y” to any prompt to overwrite an existing database.

4. The final step simply involves starting HADR on both ends, and checking it is connecting successfully. You can wait around until the standby database reaches the Peer state, or just leave it in catchup-pending, where it eventually catches up and reaches the Peer state.
  - a. In our example, the primary database never left HADR primary state, so we only need to start HADR on the standby:

```
db2 start hadr on db sample as standby
```

If HADR was not started on the primary server database, we could have achieved this by logging on there and issuing the following command:

```
db2 start hadr on db sample as primary
```

**Tip:** If HADR is stopped on both servers for a given database, restarting it should commence with the standby first, then the primary, otherwise you end up with this message:

```
SQL1768N Unable to start HADR. Reason code = "7".
```

- b. Checking that HADR is connected and working should now be a matter of checking the `db2diag.log`, or `get snapshot for db`, or `db2pd` commands.

```
db2pd -db sample -hadr
```

Output for us is as follows in Example 3-6.

*Example 3-6 Checking HADR using db2pd output*

---

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:04

HADR Information:
Role      State          SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Standby Peer          Sync          0                1381445

ConnectStatus ConnectTime          Timeout
Connected    Thu Oct 19 10:03:44 2006 (1161277424) 3

LocalHost          LocalService
polonium           DB2_HADR_2

RemoteHost         RemoteService      RemoteInstance
LEAD               DB2_HADR_1         db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000025.LOG 180      0x00000000071FC6F3

StandByFile StandByPg StandByLSN
S0000025.LOG 180      0x00000000071FC6F
```

---

- c. Optionally, after you reach the Peer state, you can switch roles. For instance, if you just restored the database on a broken server which was originally the primary system, and which is now running as standby, simply log on to that standby server and issue the non-forced takeover command:

```
db2 takeover hadr on db sample
```

In our example environment, a subsequent **db2pd** gives us the results shown in Example 3-7.

*Example 3-7 db2pd output - HADR is in Peer state*

---

```
db2inst1@polonium:/usr/db2/backup> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:32:18

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary Peer              Sync          0                  0

ConnectStatus ConnectTime              Timeout
Connected    Thu Oct 19 10:03:44 2006 (1161277424) 3

LocalHost                LocalService
polonium                  DB2_HADR_2

RemoteHost                RemoteService          RemoteInstance
LEAD                      DB2_HADR_1             db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000025.LOG 200      0x00000000072106AB

StandByFile StandByPg StandByLSN
S0000025.LOG 200      0x00000000072106AB
```

---

- d. If you are managing HADR from the DB2 Control Center, you would have cataloged remote node and database aliases, so alternatively, you can issue takeover commands remotely by specifying the remote database alias and explicit DB2 instance user ID and password. For example, if we now want to switch the roles again so that database SAMPLE on server LEAD becomes the primary, we can issue the command from the POLONIUM server:

```
db2 takeover hadr on db sam_lead user db2inst1
```

**Tip:** Do not be concerned about revealing DB2 passwords, because you are prompted for the password using an asterisk (\*) masked entry if you only specify the user.

- e. If you have only cataloged the nodes and database alias from the primary server side when initially setting up HADR, there are not any equivalent entries on the standby server. You can catalog them with the Control Center on the standby, or at the command line.

In summary, re-establishing (re-integrating) HADR should be as simple as restoring a backup from the primary over the top of the broken standby, and issuing appropriate start HADR commands, and no matter how long you take to do it, downtime is not required.





## HADR administration and monitoring

In this chapter we discuss the normal administration tasks performed in relation to a High Availability Disaster Recovery (HADR) system. We describe the methods of acquiring the HADR status, including “get snapshots”, “db2pd”, and snapshot table functions that are available in DB2 9 and later. Finally, we present a section on reading DB2’s db2diag.log file in relation to HADR.

## 4.1 Administering HADR systems

The focus of this section is to give you an insight on what is different on an HADR system as far as day-to-day DB2 administration tasks go. The administration of an HADR system is done mainly on the primary server, then replicated over to the standby. Regular database maintenance, such as backups, REORG, and RUNSTATS, are performed only on the primary. Logged operations are replicated to the standby, so it is important to understand what is a logged operation as compared to a non-logged operation.

### Replicated operations

The following operations are replicated to the standby from the primary:

- ▶ Data Definition Language (DDL):
  - Includes create and drop table, index, and more
- ▶ Data Manipulation Language (DML):
  - Insert, update, or delete
- ▶ Buffer pool:
  - Create and drop
- ▶ Table space:
  - Create, drop, or alter.
  - Container sizes, file types (raw device or file system), and paths must be identical on the primary and the standby.
  - Table space types (DMS or SMS) must be identical on both the servers.
  - If the database is enabled for automatic storage, then the storage paths must be identical.
- ▶ Online REORG:
  - Replicated. The standby's Peer state with the primary is rarely impacted by this, although it can impact the system because of the increase in the number of log records generated.
- ▶ Offline REORG: Replicated.
- ▶ Stored procedures and user defined functions (UDF):
  - Creation statements are replicated.
  - HADR does *not* replicate the external code for stored procedures or UDF object library files. In order to keep these in synchronization between the primary and the standby, it is the user's responsibility to set these up on identical paths on both the primary and the standby servers. If the paths are not identical, invocation fails on the standby.

## Non-replicated operations

The following database operations are not replicated by HADR:

- ▶ NOT LOGGED INITIALLY tables:
  - Tables created or altered with the NOT LOGGED INITIALLY option specified are not replicated. The tables are invalid on the standby and after a takeover, attempts to access these tables result in an error.
- ▶ BLOBs and CLOBs:
  - Non-logged Large Objects (LOBs) are not replicated, but the space is allocated and LOB data is set to binary zeroes on the standby.
- ▶ Data links:
  - NOT SUPPORTED in HADR.
- ▶ Database configuration changes:
  - UPDATE DB CFG is *not* replicated.
  - Dynamic database configuration parameters can be updated to both the primary and the standby without interruption. For the database configuration parameters that need a recycle of the database we recommend you follow the rolling upgrade steps. See 7.1, “DB2 fix pack rolling upgrades” on page 177.
- ▶ Recovery history file:
  - NOT automatically shipped to standby.
  - You can place an initial copy of the history file by using a primary backup image to restore the history file to the standby using the RESTORE command with the REPLACE HISTORY FILE option for example:

```
db2 restore database sample replace history file
```
  - You can also update the history file on the standby from a primary backup image by the RESTORE command for example:

```
db2 restore database sample history file
```
  - If a takeover operation occurs and the standby database has an up-to-date history file, backup and restore operations on the new primary generate new records in the history file and blend seamlessly with the records generated on the original primary. If the history file is out-of-date or has missing entries, an automatic incremental restore might not be possible; instead, a manual incremental restore operation is required.

We think that the recovery history file for the primary database is not relevant to the standby database, because it contains DB2 recovery information specific to that primary server; DB2 keeps records of which backup file to use for recovery to prior points in time, among other things.

In most HADR implementations, the standby system is only used in the primary mode as a temporary measure, while the old primary server is being fixed, making the primary version of the recovery history file irrelevant; also, because of potentially different log file numbers and locations between the primary and the standby, incorrect for use on the standby server.

For more information, refer to the following DB2 manual:

*Data Recovery and High Availability Guide and Reference*, SC23-5848.

## Replicating LOAD operations

The following LOAD operations are logged on the primary and sent to the standby:

- ▶ **LOAD COPY YES:**
  - Data is replicated on the standby if the standby can access the path or device specified in the load.
  - Otherwise, the table space is marked bad and future log records for this table space are skipped.
  - To avoid forced reintegration of the standby, you can recover the standby table space making sure that the path or devices are set correctly and re-run the LOAD command with the REPLACE option.
  - If it is not possible to have both server accessing the same path or device, you can deactivate the standby database while the load operation is performed, perform the load on the primary, place a copy of the output file in the standby path, and then activate the standby database. For this situation, also see the use of registry variable DB2LOADREC in 6.2, “DB2 HADR registry variables” on page 149.
- ▶ **NONRECOVERABLE LOAD:**
  - The equivalent table on the standby is marked bad and future log records regarding this table are skipped.
  - The standby database skips future log records that pertain to this table. You can choose to re-issue the LOAD command on the primary with the COPY YES and REPLACE options specified to bring the table back, or you can drop the table to recover the space.
- ▶ **LOAD COPY NO:**
  - Load operations with the COPY NO option specified are not supported, so the command is automatically converted to LOAD NONRECOVERABLE on the primary.
  - In order for a LOAD COPY NO option to execute on the primary you can set the DB2\_LOAD\_COPY\_NO\_OVERRIDE registry variable to COPY



YES and it converts your LOAD commands automatically into LOAD COPY YES. The DB2\_LOAD\_COPY\_NO\_OVERRIDE is ignored on the standby.

### Database backup

Database backup operations in relation to HADR are as follows:

- ▶ Backup on the standby is NOT allowed.
- ▶ Any of the standard database backups are allowed on the primary. If your goal is high availability, we recommend utilizing the online backup, in order to keep your database available.

### Database restore

Database restore operations in relation to HADR are as follows:

- ▶ Table space level restore on the primary or the standby is *not* allowed.
- ▶ Redirected restores are *not* allowed.
- ▶ A database restore can be performed on either the primary or the standby, but might reset the database to standard mode. In order to restore the database mode, the START HADR command with either the AS PRIMARY or AS STANDBY can be used.
- ▶ When restoring a primary database the old standby can be reintegrated by START HADR AS STANDBY, if the restore roll forward option is specified for a point in time at or later than the current log sequence number (LSN) on the standby.

### Log files

Database log file operations replicated in relation to archive logging are as follows:

- ▶ No replication of LOG ARCHIVES from the primary to the standby.
- ▶ Log archive occurs only on the primary to your specified archive method.
- ▶ The standby *does not* invoke USEREXIT.
  - It only extracts logs out of archive location for log replay.

## 4.2 Snapshot

The GET SNAPSHOT FOR DATABASE command is one way to get the status of the HADR databases. The snapshot can be used any time you need the current status of either the primary or the standby database.

```
db2 get snapshot for database on database name
```

The information returned represents the status of the database manager operations at the time the command is issued. HADR information is listed under the heading of HADR Status, as shown in Example 4-1.

*Example 4-1 Sample snapshot output*

---

```
HADR Status
Role                = Primary
State               = Peer
Synchronization mode = Sync
Connection status   = Connected, 10/20/2006 15:17:12.954349
Heartbeats missed   = 0
Local host          = LEAD
Local service       = DB2_HADR_1
Remote host         = polonium
Remote service      = DB2_HADR_2
Remote instance     = db2inst1
timeout(seconds)    = 3
Primary log position(file, page, LSN) = S0000029.LOG, 532,
00000000082FC4C6
Standby log position(file, page, LSN) = S0000029.LOG, 532,
00000000082FC4C6
Log gap running average(bytes) = 0
```

---

Within the HADR Status section, most of the output reflects your current settings set in the database configuration file. The settings that can have status changed are Role, State, Connection status, Heartbeats missed, Primary log position, Standby log position, and the Log gap running average.

### **Role**

This field indicates the database the snapshot is run against. The database role can be either Primary or Standby.

### **State**

The status for HADR *state* depends on which server you are running the snapshot command. A description of each state is given here:

- ▶ Possible states on the primary database are as follows:
  - Peer:  
The primary is communicating with its standby and is shipping log buffer flushes.
  - Remote catchup:  
The primary is communicating with its standby and is shipping old logs.

- Remote catchup pending:

The primary is not shipping any logs (old or new) to the standby. The standby is trying to establish a connection with the primary. While in this state, you could potentially have a problem with the connection between the HADR servers.

- DisconnectedPeer:

The connection status is “Disconnected”. However the primary database is still in the peer window phase, therefore, the update transactions are suspended until the end of peer window phase. The end time timestamp is shown in PeerWindowEnd field of the db2pd report.

- ▶ Possible states on the standby database are as follows:

- Peer:

Caught up to the tail of the log and is receiving currently-generated log data from the primary and replaying it.

- Remote catchup:

Not caught up to the tail of the log, has requested old log data from the primary that is not in the standby’s log path and is not available through the standby’s userexit.

- Remote catchup pending:

Finished its local catchup phase and is ready to receive log data from the primary. While in this state, you could potentially have a problem with the connection between the HADR servers.

- Local catchup:

Performing log replay from log data already on the disk of that system.

- DisconnectedPeer:

The connection status is “Disconnected”. However the standby database is still in the peer window phase. The **TAKEOVER HADR . . . PEER WINDOW ONLY** command can run successfully during this phase.

### ***Connection status***

The connection status returns the current connection status between the primary and the standby. This can be one of the following three possibilities:

- ▶ Connected:

The database is connected to its partner node.

- ▶ Disconnected:

The database is not connected to its partner node. If the disconnected status is not expected, you should check the status of the other HADR server either

the primary or the standby, to determine if there is a database problem, or some network issue.

► **Congested:**

The database is connected to its partner node, but the connection is congested. With a congested status, we suggest that you troubleshoot your network to pinpoint the problem for the congestion.

***Heartbeats missed***

Heartbeats are used by the HADR pair to check each other's status. If this number is zero, all heartbeats have been received. The higher this value the worse the connection condition is in. An HADR database expects at least one heartbeat message from the other database for each quarter of the time interval defined in the HADR\_TIMEOUT configuration parameter, or 30 seconds, whichever is shorter.

For example, if your HADR\_TIMEOUT is set to 40 seconds, the HADR database expects one heartbeat every 10 seconds. If no heartbeats are received in an HADR\_TIMEOUT interval, the partner is considered not responding and the connection is closed. In the case of a closed connection you must determine the cause; either the other HADR server is having problems or there are network problems. After the problem is found and fixed, you can use the START HADR command to re-establish the HADR communication, if the logs are synchronized, otherwise you must re-establish HADR through a restore.

***Primary and standby log position***

These elements show the status on the log position of both the primary and the standby. Because log positions are transmitted from the partner node periodically, these may not be accurate. If log truncation takes place the “log gap running average” would not be accurate.

***Log gap running average***

This element shows the running average of the gap between the primary LSN and the standby LSN. The gap is measured in number of bytes. Because this is an average number, the value might not be 0 even the systems are in SYNC mode and in Peer state.

## 4.3 Monitoring HADR: db2pd

The current state of HADR can be gathered without explicit connection to a database using the **db2pd** command. Note that this command requires SYSADM authority in DB2, and also requires that the database be activated in order to get any useful output related to HADR.

Example 4-2 shows the command syntax and *monitor element* output for the **db2pd** command to get HADR information for the SAMPLE database on our HADR primary server (Zaire).

*Example 4-2 Output of db2pd on the primary server (Zaire)*

---

```
$ db2pd -d sample -hadr
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:05:46
```

```
HADR Information:
```

Role	State	SyncMode	HeartBeatsMissed	LogGapRunAvg (bytes)
Primary	Peer	Nearsync	0	0

ConnectStatus	ConnectTime	Timeout
Connected	Wed May 7 11:57:56 2008 (1210186676)	30

PeerWindowEnd	PeerWindow
Wed May 7 12:06:54 2008 (1210187214)	200

LocalHost	LocalService
Zaire_int	hadrs_db2inst1

RemoteHost	RemoteService	RemoteInstance
Baltic_int	hadrp_db2inst1	db2inst1

PrimaryFile	PrimaryPg	PrimaryLSN
S0000008.LOG	1858	0x000000001534AD04

StandByFile	StandByPg	StandByLSN
S0000008.LOG	1858	0x000000001534AD04

---

Example 4-3 shows the output for the **db2pd** command to get HADR information for the SAMPLE database on our HADR standby server (Baltic).

*Example 4-3 Output of db2pd on the standby server (Baltic)*

---

```
$ db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:12:56

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Standby Peer          Nearsync 0                0

ConnectStatus ConnectTime              Timeout
Connected    Wed May  7 11:57:04 2008 (1210186624) 30

PeerWindowEnd              PeerWindow
Wed May  7 12:06:54 2008 (1210187214) 200

LocalHost                  LocalService
Baltic_int                 hadrp_db2inst1

RemoteHost                 RemoteService           RemoteInstance
Zaire_int                  hadrs_db2inst1         db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000008.LOG 1858      0x000000001534AD04

StandByFile StandByPg StandByLSN
S0000008.LOG 1858      0x000000001534AD04
```

---

As expected, the db2pd outputs on the HADR primary and the standby have differences and similarities. In our example (see Example 4-2 on page 107 and Example 4-3 above), the only noticeable differences are — in addition to Zaire having *HADR Role* of Primary and Baltic having Standby — that the *LocalHost*, *LocalService* monitor element values are swapped with *RemoteHost* and *RemoteService*.

In a long-running HADR pair, after numerous log writes and HADR stop/start activity, the primary and the standby log information (*PrimaryFile*, *StandByFile*, *PrimaryPg*, *StandbyPg*, *PrimaryLSN*, *StandbyLSN*) can differ. Having said that, in an *HADR State* of Peer, the output for each value shown in a **db2pd** command on the primary must still match the output for that same value shown in a **db2pd** command run on the standby at that same time. That is, if the db2pd output for StandByPg on the primary was 289, then it should also be 289 for the db2pd output for StandByPg on standby, even if PrimaryPg was some other value.

You might also notice that the *LogGapRunAvg* value is often not 0 bytes, even when *HADR State* is Peer.

The *log gap running average* value actually represents the average of the difference between the LSN of the primary and the standby, at the last time the primary and the standby databases were communicating. If a primary log is truncated for any reason, a gap appears as the primary has moved to the start of the next log and the standby is still at the end of the last log. This gap should close after the next log write by the primary.

The information about the peer window, *PeerWindowEnd* and *PeerWindow*, are the new output from DB2 9.5. *PeerWindowEnd* means the end time of peer windows phase. After the time specified by *PeerWindowEnd*, the primary database starts to accept the update transactions. Both the primary and standby databases have same *PeerWindowEnd* value because this value is sent from primary database with heartbeats.

For more information about db2pd and HADR monitoring, refer to the following DB2 manuals:

- ▶ *Data Recovery and High Availability Guide and Reference*, SC23-5848
- ▶ *Command Reference*, SC23-5846
- ▶ *System Monitor Guide and Reference*, SC23-5865

## 4.4 Monitoring HADR: Administrative view and table function

From DB2 V9.1 and later, a new way to view the various *monitor elements* within HADR is to use either an *administrative view*, or a *table function*.

*Monitor elements* are the individual attributes for various functional areas inside DB2 which are monitored. That is, they have counters or point-in-time status values updated as relevant events occur. Related monitor elements are referred to as *logical data groups*.

For the HADR logical data group in DB2, administrative views and table functions allow applications and scripts alike to use simple SQL queries and get a single value for a single HADR monitor element (or formatted output with multiple columns as required) rather than attempting to write complicated string manipulation routines or use **awk** and **grep**, to strip the output of **db2pd** or **db2 get snapshot for database**.

Example 4-4 and Example 4-5 show some sample syntax to effectively get the same result, specifically, the current value of the HADR\_STATE monitor element.

*Example 4-4 SQL table function to get the current primary HADR\_STATE*

---

```
> db2 "select hadr_state from table (SNAP_GET_HADR('sample',-1)) as SH"
```

```
HADR_STATE  
-----  
PEER
```

```
1 record(s) selected.
```

---

*Example 4-5 SQL Administrative View to get the current primary HADR\_STATE*

---

```
> db2 "select HADR_STATE from sysibmadm.snaphadr"
```

```
HADR_STATE  
-----  
PEER
```

```
1 record(s) selected.
```

---

Note that these are SQL queries and require a prior connection to the database. This means that with current versions of HADR, you are unable to use either administrative views or table functions to get the values of the monitor elements for a standby database. Attempts to run these commands on the standby result in the following message:

```
SQL1024N A database connection does not exist. SQLSTATE=08003
```

As you can guess, attempts to connect to a standby database result in this message:

```
SQL1776N The command cannot be issued on an HADR standby database.  
Reason code = "1".
```



The available monitor elements for the HADR logical data group are as shown in Example 4-6. A full description of each monitor element can be found in *Administrative Routines and Views*, SC23-5843.

*Example 4-6 Output of describe table SYSIBMADM.SNAPHADR*

Column name	Type schema	Type name	Length	Scale	Null
SNAPSHOT_TIMESTAMP	SYSIBM	TIMESTAMP	10	0	Yes
DB_NAME	SYSIBM	VARCHAR	128	0	Yes
HADR_ROLE	SYSIBM	VARCHAR	10	0	Yes
HADR_STATE	SYSIBM	VARCHAR	14	0	Yes
HADR_SYNCMODE	SYSIBM	VARCHAR	10	0	Yes
HADR_CONNECT_STATUS	SYSIBM	VARCHAR	12	0	Yes
HADR_CONNECT_TIME	SYSIBM	TIMESTAMP	10	0	Yes
HADR_HEARTBEAT	SYSIBM	INTEGER	4	0	Yes
HADR_LOCAL_HOST	SYSIBM	VARCHAR	255	0	Yes
HADR_LOCAL_SERVICE	SYSIBM	VARCHAR	40	0	Yes
HADR_REMOTE_HOST	SYSIBM	VARCHAR	255	0	Yes
HADR_REMOTE_SERVICE	SYSIBM	VARCHAR	40	0	Yes
HADR_REMOTE_INSTANCE	SYSIBM	VARCHAR	128	0	Yes
HADR_TIMEOUT	SYSIBM	BIGINT	8	0	Yes
HADR_PRIMARY_LOG_FILE	SYSIBM	VARCHAR	255	0	Yes
HADR_PRIMARY_LOG_PAGE	SYSIBM	BIGINT	8	0	Yes
HADR_PRIMARY_LOG_LSN	SYSIBM	BIGINT	8	0	Yes
HADR_STANDBY_LOG_FILE	SYSIBM	VARCHAR	255	0	Yes
HADR_STANDBY_LOG_PAGE	SYSIBM	BIGINT	8	0	Yes
HADR_STANDBY_LOG_LSN	SYSIBM	BIGINT	8	0	Yes
HADR_LOG_GAP	SYSIBM	BIGINT	8	0	Yes
DBPARTITIONNUM	SYSIBM	SMALLINT	2	0	Yes

The benefit of the administrative view is the simplicity of the SQL SELECT statement interface, in addition to being available for viewing from the DB2 Control Center. Figure 4-1 shows most of the column names available in the SNAPHADR administrative view, which is essentially a Control Center equivalent of a DESCRIBE TABLE SQL statement.

This was achieved by clicking the **SAMPLE** database, which we have set up in HADR mode, connecting to the database through the **Connect** option in the right pane, clicking **Views** in the left hand tree, and finding and clicking the **SNAPHADR** view (SYSIBMADM schema). Double-clicking the **SNAPHADR** view would get us a single row result, with the current values for each of the HADR monitor elements.

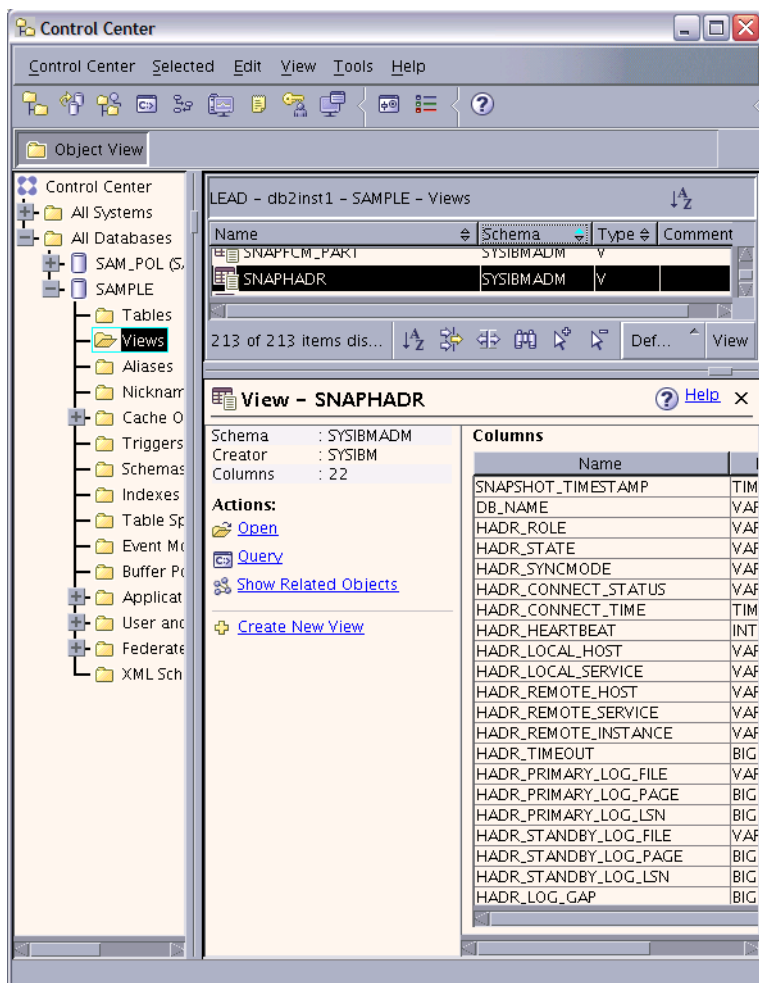


Figure 4-1 SNAPHADR administrative view as seen through DB2 Control Center

For more details about administrative views and table function, refer to the following DB2 manuals:

- ▶ *Administrative Routines and Views*, SC23-5843
- ▶ *System Monitor Guide and Reference*, SC23-5865

## 4.5 Monitoring HADR: db2diag

It is not only possible to get certain basic status information about HADR from the db2diag.log, but depending on what you need, it can be very efficient to simply awk or grep the tail of the db2diag.log file. For example, from db2diag.log, you can easily find out whether the primary server thinks it is currently in the Peer state or not. In addition, using the **db2diag** system command allows you to filter and format the output of the db2diag.log file. This neither requires overhead to achieve a connection to the DB2 database, nor interface through any API to get to the DB2 snapshot or monitor element data.

Example 4-7 and Example 4-8 show the various states of progression through HADR startup to achieve the Peer state, and the syntax of the HADR messages that you see. This can be useful whether you are just manually eyeballing the output, or if you are scanning through the db2diag.log with some string searching algorithm for specific text.

We have provided these examples specifically to show what happens in parallel time lines on the primary and the standby after an initial HADR reintegration. The db2diag.log text shown in Example 4-7 was from just after the backup file had been restored on the standby, and HADR was started, first on the standby server (POLONIUM), and then on the primary server (LEAD).

We show the events occurring on the standby first in Example 4-7, as this is where events actually commenced first.

### *Example 4-7 Progression of HADR status from startup to Peer state on the standby*

---

```
2006-10-31-14.36.55.969031-480 I563982G415      LEVEL: Warning
PID      : 13964                TID : 2988689072  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000        DB   : SAMPLE
APPHDL   : 0-976              APPID: *LOCAL.db2inst1.061031223655
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE  : Info: HADR Startup has begun.

2006-10-31-14.36.55.979794-480 E564398G310      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to None (was None)

2006-10-31-14.36.55.986878-480 I564709G333      LEVEL: Warning
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20100
MESSAGE  : HADR_LOCAL_HOST polonium mapped to 9.43.86.69

2006-10-31-14.36.55.987731-480 I565043G330      LEVEL: Warning
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20110
MESSAGE  : HADR_REMOTE_HOST LEAD mapped to 9.43.86.68
```

```

2006-10-31-14.36.55.988115-480 I565374G299      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHosts, probe:20157
MESSAGE  : HADR is using IPv4.

2006-10-31-14.36.56.010573-480 E565674G312      LEVEL: Event
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to S-Boot (was None)

2006-10-31-14.36.56.010899-480 I565987G315      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster, probe:21251
MESSAGE  : Info: Replaymaster Starting...

2006-10-31-14.36.56.012106-480 I566303G345      LEVEL: Warning
PID      : 13934          TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE  : Info: HADR Startup has begun.

2006-10-31-14.36.56.012967-480 I566649G332      LEVEL: Warning
PID      : 13934          TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE  : Starting Replay Master on standby.

2006-10-31-14.36.56.013245-480 I566982G317      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster, probe:21252
MESSAGE  : Info: Replaymaster request done.

2006-10-31-14.36.56.013480-480 E567300G322      LEVEL: Event
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to S-LocalCatchup (was S-Boot)

2006-10-31-14.36.56.013664-480 I567623G294      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20341
MESSAGE  : Info: Standby Started.

2006-10-31-14.36.56.013899-480 I567918G419      LEVEL: Warning
PID      : 13964          TID : 2988689072  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB   : SAMPLE
APPHDL   : 0-976          APPID: *LOCAL.db2inst1.061031223655
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21152
MESSAGE  : Info: HADR Startup has completed.

2006-10-31-14.36.56.020021-480 I568338G402      LEVEL: Severe
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
MESSAGE  : Failed to connect to primary. rc:
DATA #1 : Hexdump, 4 bytes
0xBF968D8 : 1900 0F81                                     ....

2006-10-31-14.36.56.020398-480 I568741G346      LEVEL: Severe

```

```

PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
RETCODE : ZRC=0x810F0019=-2129723367=SQLQ_CONN_REFUSED "Connection refused"

2006-10-31-14.36.56.014136-480 E569088G348      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000         DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE  : ADM1602W Rollforward recovery has been initiated.

2006-10-31-14.36.56.315699-480 E569437G391      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000         DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
MESSAGE  : ADM1603I DB2 is invoking the forward phase of the database
           rollforward recovery.

2006-10-31-14.36.56.316070-480 I569829G464      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000         DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
DATA #1 : <preformatted>
Invoking database rollforward forward recovery,
lowtranlsn 00000000088B800C in log file number 31
minbufflsn 00000000088B800C in log file number 31

2006-10-31-14.36.56.332364-480 I570294G374      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000         DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
DATA #1 : <preformatted>
Using parallel recovery with 3 agents 4 QSets 12 queues and 2 chunks

2006-10-31-14.36.56.338124-480 I570669G323      LEVEL: Info
PID      : 20284                TID : 2988689072  PROC : db2lfr (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgPostLogMgrToRetrieve, probe:1050
DATA #1 : <preformatted>
RTStatus is in state 16 at index 0.

2006-10-31-14.36.56.338917-480 I570993G322      LEVEL: Warning
PID      : 14923                TID : 2988689072  PROC : db2shred (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000         DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpshrEdu, probe:18300
MESSAGE  : Maxing hdrLCUEndLsnRequested

2006-10-31-14.36.56.423126-480 E571316G338      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-RemoteCatchupPending (was S-LocalCatchup)

2006-10-31-14.37.16.046636-480 E571655G346      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to S-RemoteCatchupPending (was S-RemoteCatchupPending)

2006-10-31-14.37.16.066181-480 E572002G339      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000

```

```

FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-RemoteCatchup (was S-RemoteCatchupPending)

2006-10-31-14.37.16.066431-480 I572342G312          LEVEL: Warning
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSPrepareLogWrite, probe:10260
MESSAGE : RCUStartLsn 000000000888800C

2006-10-31-14.37.16.692591-480 E572655G329          LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-NearlyPeer (was S-RemoteCatchup)

2006-10-31-14.37.16.801856-480 E572985G320          LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-Peer (was S-NearlyPeer)

```

---

To sum up the states of progression on the standby:

- ▶ *None*: HADR has not yet started, the state is set to None.
- ▶ *S-Boot*: This is the initial phase.
- ▶ *S-LocalCatchup*: This indicates that any data from the standby's log path sitting in the log replay queue is being processed.
- ▶ *S-RemoteCatchupPending*: HADR has reached the end of the local logs in the replay queue, and is now waiting for a connection to the primary in order to see if there are further logs yet to be sent across for replay.
- ▶ *S-RemoteCatchup*: HADR has made a connection to the primary, and commences receiving active/archive logs from the primary log path and replaying those logs in order to get closer to the Peer state (as opposed to Peer state, where logs are sourced from the memory on the primary).
- ▶ *S-NearlyPeer*: HADR has finished replaying all logs on the standby local disk, and is waiting for the primary to suspend log writing, and complete reading any primary on-disk logs for processing on the primary.
- ▶ *S-Peer*: From the HADR standby's point of view, it has achieved the Peer state with the primary, according to the rules of the current SYNCMODE. The source of logs in the Peer state is the primary log buffer. That is, in-memory logs, as opposed to the primary on-disk source of logs in the case of RemoteCatchup.

These states which an HADR standby database goes through, from HADR START and database activation through to achieving the Peer state, are also described in Chapter 3 of the DB2 V9.5 *Data Recovery and High Availability Guide and Reference*, SC23-5848. The output in the db2diag.log file describes internal states. There is also a prefix to show whether the state is for a primary (P-) or for a standby (S-) database. Correspondingly, this summary of states for the standby can be applied to the same states in Example 4-8 for the primary.

Defining the difference between the internal and the external states can be simplified as meaning the difference between the following states:

- ▶ The state of HADR at a lower, more accurate internal level, updated at a very high rate inside the db2diag.log file, with indicators of the primary and standby role in the state value
- ▶ The states displayed by the snapshot, db2pd, and table functions

While the rendition of external states is less granular, it is still effective for the purpose of monitoring HADR, and the `hdrMonitorState()` API function is able to map internal to external equivalent state.

Example 4-8 shows the equivalent actions occurring on the primary in the same parallel time line.

*Example 4-8 Progression of HADR status from Startup to Peer state on the primary*

---

```

2006-10-31-14.37.15.358606-480 I471198G393      LEVEL: Warning
PID       : 6585                TID  : 2988889776  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
APPHDL   : 0-125              APPID: *LOCAL.db2inst1.061031231208
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE  : Info: HADR Startup has begun.

2006-10-31-14.37.15.362854-480 E471592G310      LEVEL: Event
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to None (was None)

2006-10-31-14.37.15.372100-480 I471903G329      LEVEL: Warning
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20100
MESSAGE  : HADR_LOCAL_HOST LEAD mapped to 9.43.86.68

2006-10-31-14.37.15.372600-480 I472233G334      LEVEL: Warning
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20110
MESSAGE  : HADR_REMOTE_HOST polonium mapped to 9.43.86.69

2006-10-31-14.37.15.372779-480 I472568G299      LEVEL: Warning
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHosts, probe:20157
MESSAGE  : HADR is using IPv4.

2006-10-31-14.37.15.395413-480 E472868G312      LEVEL: Event
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-Boot (was None)

2006-10-31-14.37.15.397848-480 I473181G294      LEVEL: Warning
PID       : 29494              TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20301
MESSAGE  : Info: Primary Started.

```

```

2006-10-31-14.37.16.504640-480 E473476G330      LEVEL: Event
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-RemoteCatchupPending (was P-Boot)

2006-10-31-14.37.16.513132-480 I473807G397      LEVEL: Warning
PID      : 6585          TID : 2988889776  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
APPHDL  : 0-125          APPID: *LOCAL.db2inst1.061031231208
AUTHID  : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21152
MESSAGE : Info: HADR Startup has completed.

2006-10-31-14.37.17.133766-480 E474205G339      LEVEL: Event
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)

2006-10-31-14.37.17.144621-480 I474545G313      LEVEL: Warning
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20445
MESSAGE : remote catchup starts at 00000000088B800C

2006-10-31-14.37.17.146188-480 I474859G330      LEVEL: Warning
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrTransitionPtoNPeer, probe:10645
MESSAGE : near peer catchup starts at 00000000088B800C

2006-10-31-14.37.17.250094-480 E475190G329      LEVEL: Event
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-NearlyPeer (was P-RemoteCatchup)

2006-10-31-14.37.17.258634-480 E475520G320      LEVEL: Event
PID      : 29494          TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-Peer (was P-NearlyPeer)

```

---

## Filtering db2diag.log output

Using the unfiltered output of db2diag.log, the command shown in Example 4-9 is used to see if the primary is still in the Peer state or not. This is especially useful for the standby or cluster manager to know before issuing any **db2 takeover hadr on db ... by force** command.

*Example 4-9 Status checker script to be run continually on the HADR primary*

---

```

tail -0 -f db2diag.log | awk '
/CHANGE \: HADR state set to P-RemoteCatchupPending \(was P-Peer\) /
{system("'"$WRITE_STATUS"' 1")}
/CHANGE \: HADR state set to P-Peer \(was P-NearlyPeer\) /
{system("'"$WRITE_STATUS"' 0")}

```

---



Example 4-10 shows an example of using db2diag to filter the output of db2diag.log, with accompanying output as seen from our LEAD server.

*Example 4-10 db2diag filtering and timestamping specific output from the db2diag.log file*

```
db2diag -gi "funcname:=hdr" -fmt
"%{tsmmonth}-%{tsday}-%{tshour}.%{tsmin}.%{tssec} %{process} %{changeevent}\t
%{message}"

10-11-16.58.17 db2agent (SAMPLE) 0 Info: HADR Startup has begun.
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR state set to None (was None)
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR_LOCAL_HOST LEAD mapped to 9.43.86.68
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR_REMOTE_HOST polonium mapped to 9.43.86.69
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR is using IPv4.
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR state set to P-Boot (was None)
10-11-16.58.17 db2hadrp (SAMPLE) 0 Info: Primary Started.
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchupPending (was P-Boot)
10-11-16.58.18 db2agent (SAMPLE) 0 Info: HADR Startup has completed.
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)
10-11-16.58.18 db2hadrp (SAMPLE) 0 remote catchup starts at 000000000659000C
10-11-16.58.18 db2hadrp (SAMPLE) 0 near peer catchup starts at 000000000659000C
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-NearlyPeer (was P-RemoteCatchup)
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-Peer (was P-NearlyPeer)
10-12-09.20.18 db2hadrp (SAMPLE) 0 Zero bytes received. Remote end may have closed connection
10-12-09.20.18 db2hadrp (SAMPLE) 0 HADR primary rcv error:
10-12-09.20.18 db2hadrp (SAMPLE) 0
10-12-09.20.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchupPending (was P-Peer)
10-12-09.20.59 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)
```

For more details of monitoring HADR, refer to *Data Recovery and High Availability Guide and Reference*, SC23-5848.

## 4.6 Monitoring summary

These sections on get snapshot, db2pd, administrative views and table functions, and db2diag.log sum up the methods by which you can keep track of the HADR monitor elements, outside of the GUI interfaces to those same methods, such as the Manage HADR dialog box in the DB2 Control Center.

As you can appreciate, each method has a situation in which it is best suited:

- ▶ The get snapshot and db2pd output are similar enough to be used interchangeably, with some minor considerations. Although the get snapshot output contains all database monitor elements rather than just the HADR and as such can fill a few screens with output, it only requires SYSMON as a minimum authorization, whereas db2pd requires SYSADM.

- ▶ Administrative views and table functions with their extremely simple interface and specifically granular output, would be perfectly suited to a custom-built monitoring application for an operator flight deck. This could sit running in the background without having to run a dedicated DB2 Control Center and all the associated memory overhead. However, both these methods require a database connection and as such do not currently work on standby databases.
- ▶ Monitoring using db2diag.log requires no database connection overhead and is therefore extremely high performance oriented, it gives HADR state output for both the primary and the standby sides. Unfortunately, it requires a fair bit of string manipulation logic to get the information you need, it does not contain all the monitor element information, and is a historical point-in-time event log, not an on-demand resource. You cannot ask for the current HADR state and have it refreshed into the db2diag.log for retrieval. Monitoring must occur on a continual basis.



## Automatic client reroute

In this chapter we describe an important DB2 feature known as *Automatic client reroute* (ACR). ACR enables a DB2 client application to recover from a loss of communications so that the application can continue its work with minimal interruption.

We discuss the following topics:

- ▶ Automatic client reroute overview
- ▶ Automatic client reroute tuning
- ▶ Limitations
- ▶ Examples

## 5.1 Automatic client reroute overview

When there is a loss of communication between the client and the database server, the client receives a communication failure that terminates the application with an error. If high availability is important, you should implement a redundant setup with the ability to failover to the redundant system when the primary system fails. Automatic client reroute is a DB2 feature that enables a DB2 Client to recover from a loss of connection to the DB2 server by rerouting the connection to an alternate server. This automatic connection rerouting occurs automatically.

### 5.1.1 DB2 automatic client reroute with HADR

Figure 5-1 illustrates the automatic client reroute with HADR.

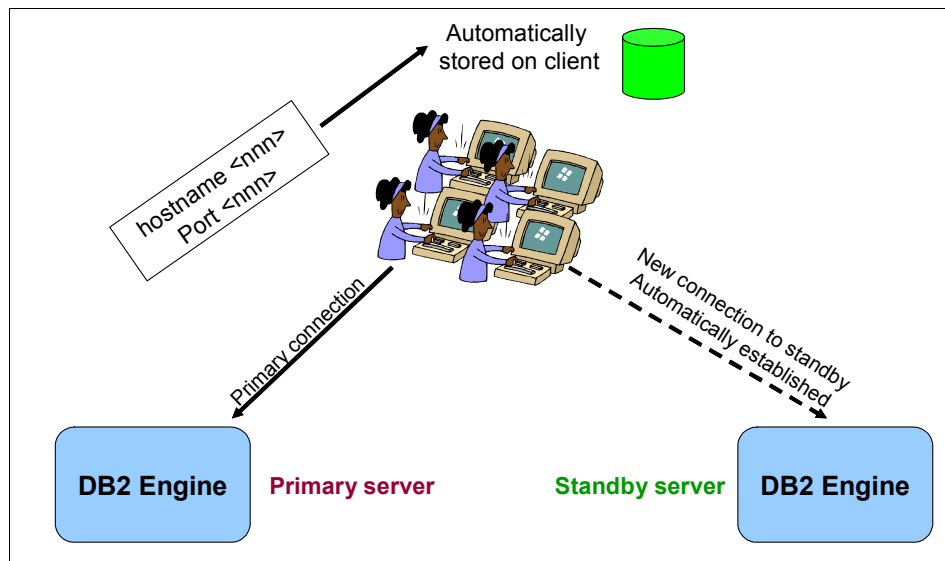


Figure 5-1 Automatic client reroute implementation in HADR

To enable the automatic client reroute feature, the DB2 server is configured with the name of an alternative location that the client can access. The UPDATE ALTERNATE SERVER FOR DATABASE command is used to define the alternate server location on a particular database.

```
db2 update alternate server for database db2 using hostname XYZ  
port yyyy
```

The alternate host name and port number is given as part of the command. The location is stored in the system database directory for the instance.

The alternate server location information is propagated to the client when the client makes a connection to the DB2 server. If communication between the client and the server is lost for any reason, the DB2 Client attempts to re-establish the connection by using the alternate server information.

The automatic client reroute feature could be used within the following configurable environments:

- ▶ Enterprise Server Edition (ESE) with the data partitioning feature (DPF)
- ▶ SQL Replication
- ▶ Q-replication
- ▶ Tivoli System Automation (TSA)
- ▶ High Availability Cluster Multiprocessor (HACMP)
- ▶ High Availability Disaster Recovery (HADR)

## 5.1.2 DB2 automatic client reroute in action

If an alternate server is specified, automatic client reroute is activated when a communication error (SQLCode -30081) or a SQLCode -1224 is detected. These error codes are not returned to the client but they activate DB2 automatic client reroute feature. In a high availability disaster recovery (HADR) environment, the automatic client reroute is also activated if SQLCode -1776 is returned.

When automatic client reroute is activated, DB2 Client code first attempts to re-establish the connection to the original server, if that fails, it tries the alternate server. The DB2 Client continues the attempt to connect with the original server and the alternate server, alternating the attempts between the two servers until it gets a successful connection for ten minutes from the moment it detected a communication failure expire whichever happens first. The timing of these attempts varies from the initial very rapid connection attempts between the two servers with a gradual lengthening of the intervals between the attempts.

*Table 5-1 Automatic client reroute connection intervals*

<b>Time</b>	<b>Interval between attempts</b>
0 to 30 seconds	0 seconds
30 to 60 seconds	2 seconds
1min to 2 minutes	5 seconds
2min to 5 minutes	10 seconds
5min to 10minutes	30 seconds

When a connection is successful, the SQLCode -30108 or SQLCode -4498 (for IBM Data Server Driver for JDBC™ and SQLJ clients) is returned to indicate that a database connection has been re-established following the communication failure. Example 5-1 shows the SQLCode 4498.

*Example 5-1 SQLCode 4498*

---

```
SQLException: com.ibm.db2.jcc.c.ClientRerouteException:  
[ibm][db2][jcc][t4][2027][11212] A connection failed but has been  
re-established. The host name or IP address is "9.43.86.111" and the  
service name or port number is 50,030.Special registers may or may not  
be re-attempted (Reason code = 1).SQLCode: -4498
```

---

Figure 5-2 shows the sequence of steps when there is a connection failure to the primary database.

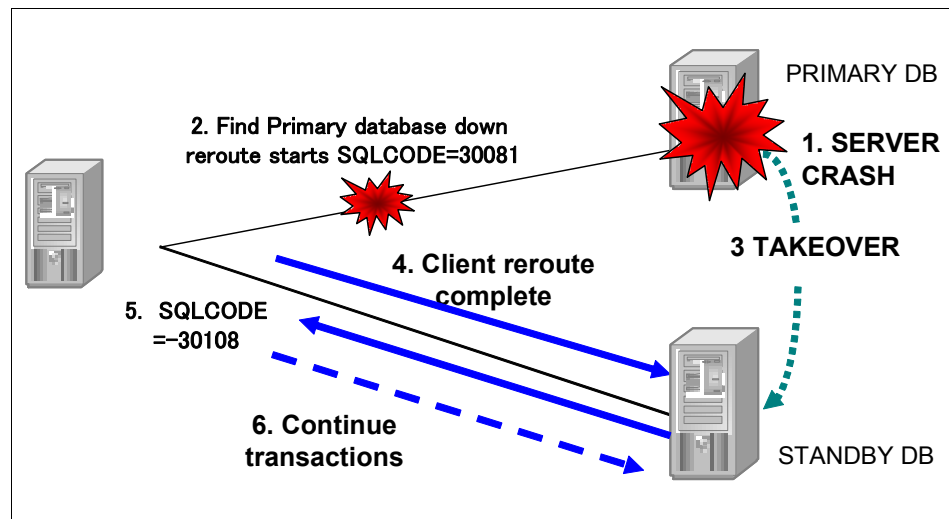


Figure 5-2 Automatic client reroute in action

## 5.2 Automatic client reroute tuning

DB2 Client actually tries to connect to the original server first before connecting to the alternate server when client reroute is triggered. Therefore, to enable fast reroute to the alternate server, the connection to the original server which has just died has to get a timeout error quickly. There are three options to achieve this:

► Tuning TCP/IP network:

To make the automatic client reroute perform better, we can tune the TCP/IP network parameters of the operating system on which DB2 application is running. For example, on AIX operating system, tune `tcp_keepinit`. The default value 150 means that 75 seconds has to be passed before the connection is really routed to the alternate server.

► Configuring DB2 registry variables:

The operating system network parameters affects all the applications on the system. So it is recommended to configure the following DB2 client reroute registry variables which affect only DB2 applications. You have to tune the parameters that enable DB2 to quickly figure out that the network connection is not alive:

– `DB2TCP_CLIENT_CONTIMEOUT`:

This registry variable specifies the number of seconds a client waits for the completion on a TCP/IP connect operation. If a connection is not established in the seconds specified, then the DB2 database manager returns the error code -30081. There is no timeout if the registry variable is not set or is set to 0.

– `DB2TCP_CLIENT_RCVTIMEOUT`:

This registry variable specifies the number of seconds a client waits for data on a TCP/IP receive operation. If data from the server is not received in the seconds specified, then the DB2 database manager returns the error code -30081. There is no timeout if the registry variable is not set or is set to 0. This variable should be set to longer than the longest query execution time. Otherwise, the client does a timeout during a long query and thinks that the connection is broken.

By default, the automatic client reroute feature retries the connection to a database repeatedly for up to 10 minutes. It is, however, possible to configure the exact retry behavior using one or both of these two registry variables:

– `DB2_MAX_CLIENT_CONNRETRIES`:

This registry variable specifies the maximum number of connection retries attempted by the automatic client reroute. If this registry variable is not set, but the `DB2_CONNRETRIES_INTERVAL` is set, the default value is 10.

– `DB2_CONNRETRIES_INTERVAL`:

This registry variable specifies the sleep time between consecutive connection retries, in number of seconds. If this variable is not set, but `DB2_MAX_CLIENT_CONNRETRIES` is set, the default value of `DB2_CONNRETRIES_INTERVAL` is 30 seconds.

If neither `DB2_MAX_CLIENT_CONNRETRIES` nor `DB2_CONNRETRIES_INTERVAL` is set, the automatic client reroute feature reverts to its default behavior.

Automatic client reroute tries to connect to both original and alternate servers in a set of trials. In one set of trials, the client connects four times as follows:

- Two times for original server
- Two times for alternate server

Client reroute waits for `DB2_CONNRETRIES_INTERVAL` after one set of trials, then repeats the trial process for `DB2_MAX_CLIENT_CONNRETRIES`. After the trial reaches `DB2_MAX_CLIENT_CONNRETRIES`, it waits for `DB2_CONNRETRIES_INTERVAL`. Then automatic client reroute fails with error code SQL30081.

For example, if the following registry variables had been set as follows:

- `DB2_MAX_CLIENT_CONNRETRIES=5`
- `DB2_CONNRETRIES_INTERVAL=3`
- `DB2TCP_CLIENT_CONTIMEOUT=10`

DB2 automatic client reroute takes  $\{(10 \times 4) + 3\} \times 5 = 215$  seconds to give up and give a communications failure error SQL30081.

**Note:** Users of Type 4 connectivity with IBM Data Server Driver for JDBC and SQLJ should use the following two DataSource properties to configure the automatic client reroute:

- ▶ **maxRetriesForClientReroute:** Use this property to limit the number of retries if the primary connection to the server fails. This property is only used if the `retryIntervalClientReroute` property is also set.
- ▶ **retryIntervalForClientReroute:** Use this property to specify the amount of time (in seconds) to sleep before retrying again. This property is only used if the `maxRetriesForClientReroute` property is also set.

For CLI/ODBC, OLE DB, and ADO.NET applications, you can set a connection timeout value to specify the number of seconds that the client application waits for a reply when trying to establish a connection to a server before terminating the connection attempt and generating a communication timeout.

If client reroute is activated, you need to set the connection timeout value to a value that is equal to or greater than the maximum time it takes to connect to the server. Otherwise, the connection might timeout and be rerouted to the alternate server by client reroute. For example, if on a normal day it takes about 10 seconds to connect to the server, and on a busy day it takes about 20 seconds, the connection timeout value should be set to at least 20 seconds.

In the `db2clini.ini` file, you can set the `ConnectTimeout` CLI/ODBC configuration keyword to specify the time in seconds to wait for a reply when



trying to establish a connection to a server before terminating the attempt and generating a communication timeout. By default, the client waits indefinitely for a reply from the server when trying to establish a connection.

If ConnectTimeout is set and client reroute is activated, a connection is attempted only once to the original server and once to the alternate server. Because the ConnectTimeout value is used when attempting to connect to each server, the maximum waiting time is approximately double the specified value for ConnectTimeout. If neither server can be reached within the time limit specified by the keyword, you receive an error message, as shown in Example 5-2.

*Example 5-2 Communication error message*

---

```
SQL30081N A communication error has been detected. Communication
protocol being used: "TCP/IP". Communication API being used:
"SOCKETS". Location where the error was detected: "<ip
address>".Communication function detecting the error: "<failing
function>". Protocol specific error code(s): "<error code>", "*",
"*". SQLSTATE=08001
```

---

When making DB2 database connections through the DB2.NET Provider, you can set the value of the ConnectTimeout property of DB2Connection when passing the ConnectionString, as shown in Example 5-3.

*Example 5-3 Set ConnectionString for DB2 .NET Provider*

---

```
String connectString =
"Server=srv:5000;Database=test;UID=adm;PWD=abd;Connect Timeout=30";
DB2Connection conn = new DB2Connection(connectString);
conn.Open();
```

---

In this example, if the connection attempt takes more than thirty seconds, the connection attempt is terminated. The setting of ConnectionString has higher priority than the registry variables or db2cli.ini setting.

Similarly, for OLE .NET Data Provider and ODBC .NET Data Provider, you can also set ConnectTimeout property by ConnectionString (Example 5-4).

*Example 5-4 Set ConnectionString for ODBC .NET Data Provider*

---

```
OleDbConnection con = new
OleDbConnection("Provider=IBMDADB2.DB2;Data
Source=TEST;ConnectTimeout=15");
```

```
OdbcConnection con = new
OdbcConnection("DSN=test;ConnectTimeout=15");
```

---

The setting of `ConnectionString` has higher priority than the settings in the registry variables or in the `db2cli.ini` file.

- ▶ Using the same cluster manager:

The third option to have fast failover is to set up an HADR pair where the primary and standby databases are serviced by the same cluster manager. Refer to example in 5.4.3, “ACR with an HADR database in HACMP cluster” on page 132 for more details. In that example, we define a service IP address for the connection from DB2 clients, and take it over when the primary fails. We use the service IP address for both “catalog tcpip node” executed on the client and “update alternate server” executed on the server. This method is the fastest but it requires an HA cluster configuration.

## 5.3 Automatic client reroute limitations

There are some limitations when using the automatic client reroute feature:

- ▶ The DB2 server installed in the alternate host server must be the same version (but could have a higher or lower FixPak) when compared to the DB2 installed on the original host server.
- ▶ If the connection is re-established to the alternate server, any new connection to the same database alias is connected to the alternate server. If you want any new connection to be established to the original location in case the problem on the original location is fixed, there are three methods to achieve it:
  - You need to take the alternate server offline and allow the connections to fail back over to the original server. This requires that the original server has been cataloged using the `UPDATE ALTERNATE SERVER` command such that it is set to be the alternate location as the alternate server.
  - You could catalog a new database alias to be used by the new connections.
  - You could uncatalog the database entry and re-catalog it again.
- ▶ The alternate server information is always kept in memory. If you do not have the authority to update the database directory (or because it is a read-only database directory), other applications are not able to determine and use the alternate server, because the memory is not shared among applications.
- ▶ The client is unable to reestablish the database connection if the alternate location has a different authentication type than the original location. The same authentication is applied to all alternate locations.
- ▶ When there is a communication failure, all session resources such as global temporary tables, identity, sequences, cursors, server options (`SET SERVER OPTION`) for federated processing and special registers are all lost. The

application is responsible to re-establish the session resources in order to continue processing the work. You do not have to run any of the special register statements after the connection is re-established, because DB2 replays the special register statements that were issued before the communication error.

However, some of the special registers are not replayed. They are:

- SET ENCRYPTPW
- SET EVENT MONITOR STATE
- SET SESSION AUTHORIZATION
- SET TRANSFORM GROUP

**Note:** If the client is using CLI, IBM Data Server Driver for JDBC and SQLJ drivers, after the connection is re-established, for those SQL statements that have been prepared against the original server, they are implicitly prepared again with the new server. However, for embedded SQL routines (for example, SQC or SQX applications), they are not prepared again.

## 5.4 Automatic client reroute configuration examples

Automatic client reroute is activated by updating the alternate server for the database on the DB2 server. Use the DB2 LIST DATABASE DIRECTORY command to confirm the alternate server as shown in Example 5-5.

*Example 5-5 Check the alternate server setting*

---

### db2 list db directory

```
System Database Directory
Number of entries in the directory = 1
Database 1 entry:
Database alias                = SAMPLE
Database name                 = SAMPLE
Local database directory      = /home/hadrinst/dbdir
Database release level        = a.00
Comment                       =
Directory entry type          = Indirect
Catalog database partition number = 0
Alternate server hostname      = 9.43.86.111
Alternate server port number   = 50030
```

---

The client applications receive SQLCode -30108 to indicate that the connection has been re-established. The transaction that was executing when the communications failure occurred is in an unknown state. The client application

has to be designed such that it can ignore this error and proceed to the next transaction, or it could retry this transaction after confirming that it has not succeeded.

Let us see how automatic client reroute works in the following cases:

- ▶ Automatic client reroute with a non-HADR database
- ▶ Automatic client reroute with an HADR database
- ▶ Automatic client reroute with an HADR database in an HACMP cluster

## 5.4.1 ACR with a non-HADR database

In this example, we use two DB2 host systems KANAGA and ATLANTIC. At KANAGA, a database SAMPLE is created. Furthermore, the database SAMPLE is also created at the alternate server ATLANTIC with port number 50030.

At the client machine, the database SAMPLE is cataloged at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at server KANAGA as follows:

```
db2 update alternate server for database sample using hostname atlantic
port 50030
```

Without having the automatic client reroute feature set up, if there is a communication error when executing a transaction, the application receives an error SQL30081N. With the automatic client reroute feature set up, DB2 tries to establish the connection to host KANAGA again. If it is still not working, DB2 tries the alternate server location (host ATLANTIC with port 50030). Assuming that there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or resubmit the transaction if it has failed. Because we do not have HADR in this case, we have to consider how to keep the two databases in sync and the standby to be online when the primary is down.

From the client machine, if we connect to the database SAMPLE and run a SELECT query as in Example 5-6, we see that the queries are run at KANAGA.

*Example 5-6 Run queries at primary DB2 system*

---

```
$db2 select id from staff fetch first 1 row only
ID
-----
    10

    1 record(s) selected.
```

```
$ db2 "select substr(host_name,1,8) from table(db_partitions()) as t"
1
-----
kanaga.i

1 record(s) selected.
```

---

If there is a communications error while executing the same query to select one row from the STAFF table, the automatic client reroute function routes the query to the alternate server. See Example 5-7.

*Example 5-7 Connection has been reroute*

---

```
$db2 select id from staff fetch first 1 row only
SQL30108N A connection failed but has been reestablished.
The hostname or IP address is
"192.168.1.20" and the service name or port number
is "50030". Special registers may or may not be reattempted
(Reason code = "1"). SQLSTATE=08506
```

```
$db2 select id from staff fetch first 1 row only
ID
-----
10

1 record(s) selected.
```

```
$ db2 "select substr(host_name,1,8) from table (db_partitions()) as t"
1
-----
atlantic.i

1 record(s) selected.
```

---

## 5.4.2 ACR with an HADR database

In this example, we set up an HADR configuration between KANAGA and ATLANTIC for database SAMPLE. At KANAGA, a primary database SAMPLE is created. A standby HADR database is also created at host ATLANTIC with port 50030.

At the client machine, a database SAMPLE is cataloged at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at KANAGA as follows:

```
db2 update alternate server for database sample using hostname atlantic  
port 50030
```

To enable an alternate server for the standby server at ATLANTIC which can fail back to primary server at KANAGA port 50030 which can be reintegrated into the HADR pair, you need to run this command on host ATLANTIC.

```
db2 update alternate server for database sample using hostname kanaga  
port 50030
```

Without having the automatic client reroute feature set up, if there is a communication error, the application receives an error SQL30081N. When the primary database server fails, applications receive an error SQL30081N even if HADR successfully brings up the standby server.

If the automatic client reroute feature is set up, DB2 tries to establish the connection to host KANAGA again. If it is still not working, DB2 tries the alternate server location (host ATLANTIC with port 50030). Assuming that there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or retry only the failed transactions. The alternate server location is where the HADR standby is located. For client reroute to succeed, HADR has to successfully bring up the standby server as the new primary. This process has to be initiated through the HADR takeover command either manually or by automating this takeover through some other HA software like HACMP.

### 5.4.3 ACR with an HADR database in HACMP cluster

In this scenario, we set up an HADR pair on KANAGA and ATLANTIC where the primary and standby databases are serviced by the same cluster manager. HADR is established between KANAGA and ATLANTIC for database SAMPLE. At the server KANAGA, primary database SAMPLE is created. A standby HADR database is also created at host ATLANTIC with port 50030.

HACMP software detects when the active node is down and do the IP takeover as well as issue the HADR takeover command to make the standby as the new primary server. In this case the HACMP service IP address is configured as 9.43.86.111.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at KANAGA using the service IP as the host name as follows:

```
db2 update alternate server for database sample using hostname  
9.43.86.111 port 50030
```

To enable an alternate server for the standby server ATLANTIC, which can fail back to primary server at KANAGA port 50030, you need to run the following command on host ATLANTIC. Again, the service IP is used as the host name.

```
db2 update alternate server for database sample using hostname  
9.43.86.111 port 50030
```

After failback to the old primary server, the old primary server (KANAGA) should be reintegrated into the HADR pair.

At the client machine, the database SAMPLE is cataloged at service IP address 9.43.86.111 at port 50030.

In this HADR with HACMP environment, service IP address (9.43.86.111) was used as the alternate server on both the primary and the standby servers. The service IP address (9.43.86.111) was also used as the host name when cataloging the database SAMPLE at the client. If the primary database server at KANAGA goes down, DB2 keeps trying to establish the connection to host service IP address 9.43.86.111.

Initially the service IP owner is KANAGA. When the HACMP detects that the primary node KANAGA is down, it performs takeover and the standby node ATLANTIC owns the resource. DB2 continues trying to connect to the service IP at alternate server location (host 9.43.86.111 with port 50030). Assuming that HACMP has successfully done the IP takeover and moved the resource group to ATLANTIC and there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or retry the failed transactions.

In an HACMP/HADR cluster, we could also have configured the alternate servers similar to the “ACR with an HADR database” on page 131. This is another option to configure the alternate servers in an HACMP/HADR cluster.

At the client machine, catalog the database SAMPLE at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

At host KANAGA, which is the primary HADR database, the alternate server is specified using the host name instead of service IP:

```
db2 update alternate server for database sample using hostname atlantic  
port 50030
```

At host ATLANTIC, which is the standby HADR server, the alternate server name is also specified using host name instead of service IP:

```
db2 update alternate server for database sample using hostname kanaga  
port 50030
```

Using the service IP address as the alternate server configuration is faster because HACMP detects the node is down and does an IP fail over. When alternate client reroute detects a communications failure, it first tries to connect to the original server. There is a good chance that the HACMP IP takeover and resource group transition has succeeded, and DB2 could get a successful connection on the original server address (service IP address 9.43.86.111), which is now transitioned to host ATLANTIC.

HACMP can make it all so seamless that sometimes when the client machine has cataloged the database at a node using the Service IP address, it is not clear which host machine is currently the primary server. To find out the primary database server host name, you can execute the following query:

```
db2 select substr(host_name,1,8) from table (db_partitions()) as t
```

## 5.5 Application programming to handle automatic client reroute

In this section we show how to handle the Automatic client reroute in application programming. An embedded SQL C application and Java application code samples are provided.

### 5.5.1 Java application with JDBC driver

The IBM Data Server Driver for JDBC and SQLJ supports client reroute for connections that use the *javax.sql.DataSource*, *javax.sql.ConnectionPoolDataSource*, *javax.sql.XADataSource*, or *java.sql.DriverManager* interface in DB2 9.5. The IBM Data Server Driver for JDBC and SQLJ supports two types of connectivity, Type 2 and Type 4. Both connectivity types support client reroute.

Client reroute for a Java application connected to a DB2 server operates in the following way:

- ▶ The IBM Data Server Driver for JDBC and SQLJ obtains primary and alternate server information. For the first connection:
  - If the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` properties are set on the `DriverManager` interface, the IBM Data Server Driver for JDBC and SQLJ loads those values into memory as the alternate server values, along with the primary server values `serverName` and `portNumber`.



- If the `clientRerouteAlternateServerName` and `clientRerouteAlternatePortNumber` properties are *not* set, and a JNDI store is configured by setting the property `clientRerouteServerListJNDIName` on the `DB2BaseDataSource`, the IBM Data Server Driver for JDBC and SQLJ loads the primary and alternate server information from the JNDI store into memory.
- If the `DriverManager` and `DataSource` properties are *not* set for the alternate servers, and JNDI is not configured, the IBM Data Server Driver for JDBC and SQLJ checks DNS tables for primary and alternate server information. If DNS information exists, the IBM Data Server Driver for JDBC and SQLJ loads those values into memory.
- If primary or alternate server information is *not* available, a connection cannot be established, and the IBM Data Server Driver for JDBC and SQLJ throws an exception.

For subsequent connections, the IBM Data Server Driver for JDBC and SQLJ obtains primary and alternate server values from driver memory.

**Note:** In DB2 9.5, Domain Name System (DNS) is supported as a repository of alternate server information. For client reroute during connections to DB2 Database for Linux, UNIX, and Windows servers, you can use DNS instead of the JNDI directory as a repository of alternate server information.

You can specify multiple IP addresses in a DNS entry. For client reroute, you can specify two: one for the primary server and one for the secondary server. If JNDI is not configured, the IBM Data Server Driver for JDBC and SQLJ uses the DNS addresses to identify the servers for client reroute.

- ▶ The IBM Data Server Driver for JDBC and SQLJ attempts to connect to the data source using the primary server name and port number. If the connection to the primary server fails, the client reroute acts as described in 5.1.2, “DB2 automatic client reroute in action”.

## Implementing ACR on the `DataSource` interface with JDBC

In a recent J2EE™ environment, we use the `DataSource` interface with JDBC. We explain how to set up the `javax.sql.DataSource` or `javax.sql.ConnectionPoolDataSource` interface with client reroute.

Example 5-8 shows an example of how we get a connection using a `DataSource` property file. `DB2SimpleDataSource` is one of the DB2 implementations of the `DataSource` interface.

*Example 5-8 Obtaining a database connection using DataSource interface*

---

```
DB2SimpleDataSource dataSource = new DB2SimpleDataSource();
    Properties prop = new Properties();
    FileInputStream dsPropFile = null;
    try{
        dsPropFile = new FileInputStream( DSname);
    }
    catch (FileNotFoundException fe)
    {
        System.out.println (fe.getMessage());
        throw fe;
    }
    prop.load( dsPropFile );
    dataSource.setServerName (prop.getProperty("serverName"));
    String portNum = prop.getProperty("portNumber");
    int portNo = (new Integer(portNum)).intValue() ;
    dataSource.setPortNumber (portNo);
    dataSource.setDatabaseName (prop.getProperty("databaseName"));
    dataSource.setUser (prop.getProperty("userName"));
    dataSource.setPassword (prop.getProperty("password"));
Connection con=datasource.getConnection();
con.setAutoCommit(false);
    Statement stmt = con.createStatement();
}
```

---

The best way to use a DataSource object is for your system administrator to create and manage it separately, using WebSphere or some other tool. The program that creates and manages a DataSource object also uses the Java Naming and Directory Interface (JNDI) to assign a logical name to the DataSource object. The JDBC application that uses the DataSource object can then refer to the object by its logical name, and does not need any information about the underlying data source. In addition, your system administrator can modify the data source attributes, and you do not need to change your application program.

To obtain a connection using a data source object:

1. From your system administrator, obtain the logical name of the data source to which you need to connect.
2. Create a Context object to use in the next step. The Context interface is part of the Java Naming and Directory Interface (JNDI), not JDBC.
3. In your application program, use JNDI to get the DataSource object that is associated with the logical data source name.

4. Use the `DataSource.getConnection` method to obtain the connection.

By using the `javax.sql.DataSource` interface, alternate server parameters can be picked up by the Java application and kept in non-volatile storage on the client machine. The storage can be done using the JNDI API. If, for instance, a local file system is specified as the non-volatile storage, JNDI creates a `.bindings` file, which contains the required alternate server information.

After the current JVM™ is shut down, the information then persists in that file until a new JVM is created. The new JVM attempts to connect to the server. If the alternate server information has been updated, this is updated on the client machine without requiring your intervention. If the server is missing however, the `.binding` file is read and a new connection attempt is made at the location of the alternate server. LDAP can also be used to provide non-volatile storage for the alternate server information. Using volatile storage is not recommended, as a client machine failure could result in the loss of alternate server data stored in memory.

To register the alternate server with JNDI and make it persistent:

1. Create an instance of `DB2ClientRerouteServerList`, and bind that instance to the JNDI registry. `DB2ClientRerouteServerList` is a serializable Java bean with four properties:

- `alternateServerName`
- `alternatePortNumber`
- `primaryServerName`
- `primaryPortNumber`

Getter and setter methods for accessing these properties are provided.

Example 5-9 shows the code sample of binding of an instance of `Db2ClientRerouteServerList` “address” to the JNDI registry.

*Example 5-9 Binding of `Db2ClientRerouteServerList` instance to JND*

---

```
// Create a starting context for naming operations
InitialContext registry = new InitialContext();

// Create a DB2ClientRerouteServerList object
DB2ClientRerouteServerList address = new
DB2ClientRerouteServerList();

// Set the port number and server name for the primary server
address.setPrimaryPortNumber(50030);
address.setPrimaryServerName("KANAGA.itso.ibm.com");

// Set the port number and server name for the alternate server
```

```
int[] port = {50030};
String[] server = {"ATLANTIC.itso.ibm.com"};
address.setAlternatePortNumber(port);
address.setAlternateServerName(server);
registry.rebind("serverList", address);
```

---

2. Assign the JNDI name of the DB2ClientRerouteServerList object to DataSource property clientRerouteServerListJNDIName. For example:

```
datasource.setClientRerouteServerListJNDIName("serverList");
```

If you use DataSource in WebSphere Application Server environment, this step is done in the administration console of WebSphere application Server.

Here is how the IBM Data Server Driver for JDBC and SQLJ makes DB2ClientRerouteServerList persistent. After the database administrator specifies the alternate server location on a particular database at the server instance, the primary and alternate server locations are returned back to the client at connect time.

The IBM Data Server Driver for JDBC and SQLJ creates an instance of referenceable object DB2ClientRerouteServerList and stores that instance in its transient memory.

If communication is lost, the IBM Data Server Driver for JDBC and SQLJ tries to re-establish the connection using the server information that is returned from the server. The *clientRerouteServerListJNDIName DataSource property* provides additional client reroute support at the disposal of the client.

The clientRerouteServerListJNDIName has two functions:

- ▶ Allows alternate server information to persist across JVMs.
- ▶ Provides an alternate server location in case the first connection to the database server fails.

The clientRerouteServerListJNDIName identifies a JNDI reference to a DB2ClientRerouteServerList instance in a JNDI repository for alternate server information. After a successful connection to the primary server, the alternate server information that is provided by clientRerouteServerListJNDIName is overwritten by the information from the server.

The IBM Data Server Driver for JDBC and SQLJ attempts to propagate updated information to the JNDI store after a failover if clientRerouteServerListJNDIName property is defined. If clientRerouteServerListJNDIName is specified, primary server information specified in DB2ClientRerouteServerList is used for connection. If a primary server is not specified, serverName information specified on the DataSource is used.

A newly established failover connection is configured with the original DataSource properties, except for the server name and port number. In addition, any DB2 special registers that were modified during the original connection are re-established in the failover connection by IBM Data Server Driver for JDBC and SQLJ.

### Java application handling the ACR exception

After a connection is re-established using client reroute, the IBM Data Server Driver for JDBC and SQLJ throws a java.sql.SQLException to the application with SQLCODE -4498, to indicate to the application that the connection has been automatically re-established to the alternate server.

Example 5-10 shows the Java application code handling the automatic client reroute exception by catching the exception -4498.

#### *Example 5-10 Handling ACR exception in Java*

---

```
// In retry logic you don't have to retry connection or create
statement or preparestatement again

Connection con = DriverManager.getConnection(url, "USER", "PSWD");
Statement stmt = con.createStatement();
pstmt = con.prepareStatement("SELECT NAME FROM STAFF WHERE ID = ?");

// the statements above don't have to re-tried when we get client
reroute exception

do { // while loop start
try {
// transaction logic
pstmt.setInt(1,10);
int rowsUpdated = stmt.executeUpdate(
    "UPDATE employee SET firstme = 'SHILI' WHERE empno = '10'");
con.commit();
}catch(SQLException se) { //deal with client reroute exception
    int errorcode = se.getErrorCode();
    System.out.println("SQLException: " + se);
    System.out.println("MySQLCode: " + errorcode);
    if(con != null){
    try{
        con.rollback();
    }catch(Exception e)
        { e.printStackTrace(); }
    } //endif
    if((errorcode == -30108) || (errorcode == -4498)){
```

```
        System.out.println("connection is re-established, re-executing
the failed transaction."); retry = true
    } finally { // close resources}
    } //end catch
    } while(retry)
```

---

**Note:** Starting in DB2 9.5 Fix Pack 1, the seamless failover is added to client reroute operation.

During client reroute, if a connection is in a clean state, you can use the `enableSeamlessFailover` property to suppress the `SQLException` with error code -4498 that the IBM Data Server Driver for JDBC and SQLJ issues to indicate that a failed connection was re-established.

The following conditions must be satisfied for seamless failover to occur:

- ▶ The `enableSeamlessFailover` property is set to `DB2BaseDataSource YES (1)`.
- ▶ The connection is not in a transaction. In other word, the failure must occur when the first SQL statement in the transaction is executed.
- ▶ There are no global temporary tables in use on the server.
- ▶ There are no open, held cursors.

## 5.5.2 Embedded SQL Program using C

After a connection is re-established, an embedded SQL C application receives an exception `SQLCode -30108`, to indicate to the application that the connection has been automatically re-established to the alternate server.

Example 5-11 is a pseudo C code for a client application of automatic client reroute.

*Example 5-11 Handling the ACR exception in C program*

---

```
int checkpoint = 0;
check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
    if (sqlca->sqlcode == -30081)
    { // as communication is lost, terminate the application right away
      exit(1);
    } else
    { // print out the error
      printf(...);
    }
    if (sqlca->sqlcode == -30108)
```

```

{ // connection is re-established, re-execute the failed transaction
if (checkpoint == 0)
{ goto checkpt0; }
else if (checkpoint == 1)
{ goto checkpt1; }
else if (checkpoint == 2)
{ goto checkpt2; }
....
exit;
}
}
}
main()
{
connect to mydb;
check_sqlca("connect failed", &sqlca);
checkpt0:
EXEC SQL set current schema XXX;
check_sqlca("set current schema XXX failed", &sqlca);
EXEC SQL create table t1...;
check_sqlca("create table t1 failed", &sqlca);
EXEC SQL commit;
check_sqlca("commit failed", &sqlca);
if (sqlca.sqlcode == 0)
{ checkpoint = 1; }
checkpt1:
EXEC SQL set current schema YYY;
check_sqlca("set current schema YYY failed", &sqlca);
EXEC SQL create table t2...;
check_sqlca("create table t2 failed", &sqlca);
EXEC SQL commit;
check_sqlca("commit failed", &sqlca);
if (sqlca.sqlcode == 0)
{ checkpoint = 2; }
...
}

```

---







## HADR best practices

In this chapter we describe the DB2 configuration parameters and registry variables that affect HADR. We also discuss the important factors to consider when implementing an HADR solution to obtain the most optimum HADR performance.

We cover the following topics:

- ▶ DB2 HADR configuration parameters
- ▶ DB2 HADR registry variables
- ▶ Recommendations and considerations

## 6.1 DB2 HADR configuration parameters

Here we explore the relevant *database manager configuration parameters* (dbm cfg parm) and *database configuration parameters* (db cfg parm) that have some appreciable impact on HADR.

In most cases, it is not necessary to change these parameters manually after setting up HADR. However, it is always useful to know what to change and why, in the event that an architectural requirement is changed further down the track.

We do not provide specific performance recommendations in this section, because each environment requires tuning to match its own unique characteristics. Having said that, the information provided here, in the DB2 manual pages, and other references listed, are enough to give you a good idea of which settings can meet your requirements.

One thing to keep in mind, with configuration parameters, and the registry and environment variables in the next section, is that they can only be recognized and used while the database is in either the *primary*, or *standby* role, and they are ignored otherwise. Still, it is important for those parameters/variables to be set and maintained on both servers of the HADR pair, for when the time comes to switch roles or perform a forced takeover.

### 6.1.1 Basic configuration parameters

Details of the configuration parameters can be found in these DB2 manuals:

- ▶ DB2 V8.2:  
*DB2 V8.2 Information Center and Administration Guide: Performance V8*, SC09-4821-01.
- ▶ DB2 9:  
DB2 9 Information Center (**Reference** → **Configuration Parameters** → **Database Systems**) and in *DB2 9 Performance Guide* SC10-4222: Appendix A.
- ▶ DB2 9.5:  
DB2 9.5 Information Center (**Database fundamentals** → **Configuring** → **Configuration parameters** → **Database configuration parameters**)

## Configuration parameters

The following parameters are available:

▶ **AUTORESTART:**

Consider setting this db cfg parm value to OFF when using automatic client reroute, so that a broken primary server does not come back online and restart in HADR primary role, causing a split-brain scenario.

In a non-HADR environment, it is good to leave AUTORESTART set to ON, as the database will automatically activate when the instance is started and the first application connection attempt is made, attempting log/crash recovery as required for any inflight/indoubt transactions that existed at the time the database was deactivated.

When AUTORESTART is left OFF, connection attempts receive an SQL1015N message.

▶ **LOGINDEXBUILD:**

This db cfg parm value should be set to ON so that maintenance operations on indexes, such as create, or REORG, on the primary database is logged and carried out to match on the standby database.

▶ **LOGRETAIN / LOGARCHMETHn:**

You must have at least one db cfg parm set in order to use HADR. Circular logging is not supported.

LOGRETAIN is a legacy parameter from the days before a native archive logging method (LOGARCHMETHn) was supported. You can still use it to specify that DB2 active logs and archive logs should be stored together in one single location. The log file location is specified by LOGPATH and can be changed by updating NEWLOGPATH and deactivate/activate database commands.

LOGARCHMETHn (1,2) specifies the destination for the primary and mirror archive logs, respectively. This can be a disk, TSM, userexit, or vendor supplied destination driver. Having a value here means that after an active log is closed by DB2 (no more active transactions referring to it), it is moved out of the active log directory and sent to the specified target/destination.

▶ **HADR\_DB\_ROLE:**

This db cfg parm is configurable indirectly through HADR commands, to STOP, START, and TAKEOVER HADR. The possible values are:

– **STANDARD:**

The database is not in an HADR role and can be processed normally.

– PRIMARY:

The database is the HADR primary, all client interactions with the HADR pair occur here.

– STANDBY:

The database is the HADR standby, any client interaction attempts will receive an SQL1776N message. Only DB2's log replay engine is allowed to run against this database. You can issue the following command against an HADR standby database to see the log replay application:

```
db2 list applications all show detail
```

▶ HADR\_LOCAL\_HOST:

This db cfg parm refers to the host name or IP address for the current server. The reason for using a separate host field for HADR is to support the use of HADR specific network card, which is addressed by its own IP address or host name that is different from the “usual” name of the host.

▶ HADR\_LOCAL\_SVC:

Logically coupled with the above parameter, this db cfg parm specifies the port number or service name used for the local HADR component of the HADR database. Note that this is completely separate from the port/service assigned to a given DB2 Instance. The HADR service/port number is specific to each HADR database.

▶ HADR\_REMOTE\_HOST:

This db cfg parm specifies the host name or IP address of the remote server for the HADR paired databases.

▶ HADR\_REMOTE\_INST:

This db cfg parm specifies the DB2 Instance name on the remote server for the HADR paired databases. Note that there is no equivalent db cfg parm for the local server — it can be derived from the current instance environment variable.

▶ HADR\_REMOTE\_SVC:

Combined with the HADR\_REMOTE\_HOST and HADR\_REMOTE\_INST, this db cfg parm specifies the port number or service name used by the HADR component for the HADR database on the remote server. As with the HADR\_LOCAL\_SVC, this value is specific to the database, and any given pair of HADR databases must have two different TCPIP port numbers.

▶ HADR\_SYNCMODE:

This is an important db cfg parameter. The setting must match on both servers in the HADR pair.

The following values are possible:

– SYNC:

In this mode, log writes are considered successful only when logs have been written to log files on the primary database and when the primary database has received acknowledgement from the standby database that the logs have also been written to log files on the standby database. The log data is guaranteed to be stored on both servers.

– NEARSYNC:

In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and when the primary database has received acknowledgement from the standby system that the logs have also been written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site has not been transferred to nonvolatile storage with all of the log data that it has received.

– ASYNC:

In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and have been delivered to the TCP layer of the primary system's host machine. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby.

▶ HADR\_TIMEOUT:

This db cfg parm specifies the time in seconds which the DB2 HADR EDU will wait for any response from its HADR partner, before considering communication to have failed and closing connection. If HADR was in Peer state before closing connection, the primary would no longer follow Peer state semantics when HADR connection is closed. The response could be either heartbeat or acknowledgement (ACK) signals. Note that there is no timeout for ACK signal wait, and ACK signals are not even used for ASYNC mode. This value must also match on both sides of the HADR pair.

▶ HADR\_PEER\_WINDOW:

This db cfg parm specifies how long the primary database suspends the update of transaction after HADR connect state has been changed to disconnect. This parameter value must be the same between the primary and standby databases. If HADR\_SYNCMODE is set to ASYNC or HADR\_PEER\_WINDOW is set to "0", DB2 ignores this parameter. When you close the HADR connection by deactivate database command, the HADR state is changed to disconnected immediately.

► INDEXREC:

Index recreation time is both a dbm and db cfg parm that specifies when, if necessary, DB2 will attempt to rebuild invalid indexes — and specifically for HADR, whether this will occur during HADR log replay on the standby. The dbm cfg parm is meant to be used as the default for all databases for that instance. The db cfg parm, when not set to SYSTEM, will override that value.

Possible values are:

– SYSTEM:

Applies to db cfg only; accept the value for INDEXREC in the dbm cfg.

– ACCESS:

Rebuild invalid index when the index is first accessed, and then rebuild on HADR standby during log replay.

– ACCESS\_NO\_REDO:

Rebuild invalid index when index is first accessed, but leave invalid on HADR standby; no rebuild during log replay. The index would be rebuilt after an HADR takeover and the first access of the underlying table.

– RESTART:

This is the default value. Rebuild of invalid index occurs after a RESTART DATABASE or HADR takeover on the normal or primary database, and on the standby database during log replay. The AUTORESTART db cfg parm effectively means that RESTART DATABASE is implicitly issued at the time an application attempts to connect to the database. Indexes are rebuilt asynchronously (no wait) at takeover time, and synchronously (wait) at restart time.

– RESTART\_NO\_REDO:

Rebuild of invalid index occurs after a RESTART DATABASE on the normal or primary database, but not on the standby during HADR replay. Rebuild would occur only after HADR takeover.

► SELF\_TUNING\_MEM:

From DB2 9 onwards, DB2 is able to dynamically assign memory for buffer pools, package cache, lock list, sort heap, and database shared memory, using this db cfg parm. Self Tuning Memory, even when set up on both servers, can only be active on an HADR primary database.

► LOGFILSIZ:

This db cfg parm, which specifies the size of active logs, is taken from the primary and used by the standby so that the size of active log files on both servers match — the standby db cfg parm value of LOGFILSIZ is ignored.

This value is kept even after an HADR role switch or takeover, until such time as the database is restarted, when the local LOGFILSZ value is used for active log file size.

## 6.1.2 Automatic client reroute configuration parameters

Strictly speaking, this minute subset of DB2 configuration parameters is not part of HADR, or even a configuration parameter, but it is used by ACR.

The command `db2 update alternate server for database` sets values stored in the system database directory file for the server and DB2 Instance port number. Here is an example, for database SAMPLE, where our current primary server is named LEAD, and the alternate would be named POLONIUM, both using TCP/IP port 50001 for the DB2 instance DB2INST1:

```
db2 update alternate server for database sample using hostname POLONIUM
port 50001
```

The mechanism of automatic client reroute, and how it is a separate entity to HADR, are explained in Chapter 5, “Automatic client reroute” on page 121.

## 6.2 DB2 HADR registry variables

In this section we examine the DB2 registry and operating system shell environment variables pertaining to HADR.

The following information is from the DB2 9 Information Center (**Reference** → **Configuration Parameters** → **Database Systems**) as well as the *DB2 9 Performance Guide*, SC10-4222: Appendix B. The same information is also provided in the *DB2 V8.2 Information Center and Administration Guide: Performance V8*, SC09-4821-01.

► **DB2\_HADR\_BUF\_SIZE:**

This registry variable is only recognized while the database is in the standby role. By default, it is the size of the HADR standby log receive buffer. The value of this variable is calculated as twice the value of LOGBUFSZ.

The standby replay mechanism retrieves logs directly from the log receive buffer. If the standby is slow in replaying logs, and the primary keeps sending more logs to the standby, the log receive buffer will eventually become “full”, preventing the standby from receiving more logs. Saturation of the receive buffer will cause transactions on the primary to be blocked until receive buffer has more room to receive log pages.

If the synchronization mode is ASYNC, the network pipeline from the primary to the standby will eventually fill up and the primary will not be able to send any more logs. This is called “congestion”. In ASYNC mode, network congestion and saturation of the standby's receive buffer will stall the primary log processing. Hence transactions on the primary are to be blocked until congestion is cleared, provided the standby can send heartbeat and the primary can receive the standby's heartbeat.

If the synchronization mode is SYNC or NEARSYNC, the primary is likely to be able to send out one more batch of logs after the standby receive buffer is full. This batch of logs is buffered in the network pipeline between the primary and the standby. In SYNC and NEARSYNC modes, although the primary is not likely to encounter congestion when sending logs, it will still have to wait for acknowledgement messages when the standby buffer is full. Thus the primary transaction processing will eventually be blocked in all the synchronization modes when the standby receive buffer is full.

A larger standby receive buffer size (set by registry variable DB2\_HADR\_BUF\_SIZE) will allow the standby to absorb load peaks, reducing the chance of slowing the primary down. But if the sustained throughput of the standby log processing is lower than the primary log creation rate, a larger buffer will still be filled up and the primary be slowed down.

If your primary load is uneven, consider a larger standby receive buffer to absorb the peaks. Consider that in certain scenarios, both SYNC and NEARSYNC modes see a significant benefit from a very generous increase of the DB2\_HADR\_BUF\_SIZE variable, whereas the manual might give the impression that only ASYNC mode would benefit.

We recommend that you perform basic transactional throughput testing using a larger DB2\_HADR\_BUF\_SIZE value on each of the three synchronization modes to determine the best solution for your own environment. NEARSYNC is the default mode, and in many cases will provide a solution which is closest in performance to not running HADR.

► **LOAD\_COPY\_NO\_OVERRIDE**

This variable is not specific to HADR. However, it can have either an adverse or beneficial effect on an HADR database, depending on the value you assign to it. The variable is ignored on the standby databases, but applied on either a primary or a standard role database.

By default, when a **db2 load...copy no** command is run on an HADR primary database, DB2 will convert that to NONRECOVERABLE, that table space is placed in a copy pending (read only) state, and the HADR standby will cease to match the primary. The table on the standby is marked bad, and logs from the primary will not be applied there. This state can be corrected by subsequent execution of a **db2 load...copy yes** command.



If this registry variable is set to COPY YES (with a valid copy destination as part of the syntax), the load is converted such that it will automatically be used with log replay on the HADR standby to maintain data integrity. Note that the load command would still have to be using a valid target which the HADR standby database can access, in order for this to be effective (that is, a shared drive with matching path, or a mutually accessible TSM target).

► **DB2LOADREC**

This registry variable can be used when the **db2 load...copy yes** backup image location on the HADR standby does not match the target location it was sent to from the HADR primary. This can happen when shared disk relative mapping does not match on both servers, or a method of file transfer outside DB2's control is occurring. Essentially, the value of DB2LOADREC is set to a plain text file which contains all required media information about the load copy backup image. The structure and required content of this file is fully described in the *DB2 v9 Data Recovery and High Availability Guide and Reference*, from page 165, and also in *DB2 9 Data Movement Utilities Guide and Reference*, SC10-4227, page 131.

As of DB2 9, there are no Operating System Shell environment variables used to influence the operation of HADR.

## 6.3 Recommendations and considerations

In this section we point out the important things to consider when implementing an HADR solution. There is no one perfect way of configuring HADR; everything depends on the user's requirements. If transaction performance is critical, then you might have to sacrifice HADR takeover time. If there is high logging activity and the network cannot keep up with the logging, you might have to sacrifice the transaction performance. You have to weigh the pros and cons of each of the following factors on the user requirements when tuning the HADR.

### 6.3.1 DB2 transaction performance

HADR setup could have a very slight impact on the performance of the DB2 transactions. With a stricter log shipping mode, the potential performance impact is higher. The tuning in an HADR setup is just the usual tuning of DB2 parameters in a non-HADR environment. There are no specific HADR parameters to be tuned. Each environment will require tuning to match its own unique characteristics.

Depending on the log shipping mode, the performance of the in-flight transactions on the primary can get affected when the standby goes down.

The primary database continuously polls the network. If the network or the standby machine goes down, the primary database will detect the condition as soon as the primary host machine operating system detects the condition. If the standby database crashes, but the standby machine and the network are still up, in most cases, the primary machine and therefore the primary database can detect the failure and close the connection quickly. Closing the connection will cause the primary database to change from Peer state to disconnected state, unblocking pending transactions. HADR state changes from peer to disconnected.

If the primary machine is unable to report the standby or network failure in time, DB2 will rely on HADR\_TIMEOUT to disconnect. If the primary does not receive any message from the standby for HADR\_TIMEOUT seconds (default 120 seconds) since the last heartbeat, it will change the state to disconnected.

With SYNC and NEAR SYNC log shipping mode, a commit statement for in-flight transactions does not complete until HADR state is changed to disconnected. This is because for SYNC and NEARSYNC modes, the primary has to wait for the acknowledgement. With ASYNC log shipping mode, in-flight transactions do not suffer from a network disconnection or the standby down and commit statement for them complete immediately. This is because in ASYNC mode, when the primary sends logs, the operating system network layer might return send success, but buffer the data in the network path, or return “congestion”.

For ASYNC mode, if the primary keeps sending more logs, it will eventually cause congestion. If send returns congestion, the primary transaction processing is blocked immediately in ASYNC mode also. When there is a congestion or the receive buffer is full, the standby knows that it cannot receive heartbeat from the primary anymore. It updates its heartbeat receive time and continues to send heartbeat to the primary, so the standby will not drop connection after HADR\_TIMEOUT.

On the primary side, because the primary is receiving heartbeat from the standby, the primary also will not drop connection after HADR\_TIMEOUT. Because the standby can send heartbeat and the primary can receive the standby's heartbeat, the HADR pair will stay connected. In the worst case, an unresponsive standby or network could cause the primary transactions to be blocked as long as the primary can receive heartbeat from the standby.

### **6.3.2 How to reduce the takeover time**

Customers would like to keep HADR takeover time to a minimum. By reducing the amount of transaction rollback after takeover on the standby, you can open up the database to the clients sooner.

There is not a significant difference in takeover time between the different log shipping modes. When primary fails, forced takeover is usually faster than unforced. But if the standby detects the failure and is already disconnected, unforced takeover cannot be issued because the standby is no longer in Peer state.

If you know for sure that the primary has failed and would like to failover, issue forced takeover on the standby. If the standby has not detected the failure and is still in Peer state, it will allow unforced takeover. But because the primary is down, unforced takeover will not get any response from the primary and will hang for HADR\_TIMEOUT period. So if you have determined that the primary is down, you should issue forced takeover directly.

### 6.3.3 Seamless takeover

HADR does not automatically detect a failed primary and issue a takeover. This process is manual. When the user has determined that the primary is down, the user can issue the **takeover** command. But manual takeover might not be acceptable in certain situations. In that case, you can configure HADR with a clustering software such as Tivoli System Automation (TSA), HACMP, VCS, or MSCS to make this takeover completely seamless. The cluster software detects that the primary is down and issues an HADR takeover. Refer to Chapter 11, “HADR with clustering software” on page 345 for more details on implementing this solution.

### 6.3.4 Performance implications of HADR\_TIMEOUT

If one database does not receive any message from the other one for the HADR\_TIMEOUT period, the connection is closed.

Try to avoid setting the HADR\_TIMEOUT for too long. In Peer state, if the standby is not responding, transactions on the primary can hang for the HADR\_TIMEOUT period.

Setting the HADR\_TIMEOUT too short can also impact the performance. You get a false alarm on the connection. Frequent disconnecting and reconnecting are a waste of resources. HA protection also suffers as disconnection brings the primary out of Peer state. We recommend setting it to at least 10 seconds.

### 6.3.5 Applications with high logging rate

For applications that have a very high logging activity, you have to make sure that the network is capable of transmitting that load of data. The standby database should also be powerful enough to replay the logged operations of the

database as fast as they are generated on the primary. We recommend identical primary and standby hardware.

DB2 HADR does not currently support compression of the log files before sending them to the standby in order to improve this situation. However, depending on the synchronization mode that you are using, the logging rate on the primary can be reduced to the rate of transfer on the network by choosing a more strict form of log shipping mode, such as SYNC mode, which forces the primary to wait for the standby to acknowledge that it has received the log record before it can proceed. The network bandwidth is the limit of logging rate.

In most systems, the logging capability is not driven to its limit. Some systems do not see much difference among the three synchronization modes, or with the HADR enabled or disabled. This behavior is usually seen in systems where logging is not the bottleneck of database performance.

### 6.3.6 Network considerations

The primary database ships the log records to the standby database server over the TCP/IP network when the primary does a log flush. In the primary database, SQL agent EDUs produce logs and write the logs into the database log buffer, whose size is controlled by the database configuration parameter LOGBUFSZ. The loggw EDU consumes the logs by writing them to disk. The write operation is called log flushing. Each write operation (and the logs written) is called a flush. For HADR primary databases, each flush is also sent to the standby database. For each write call, there is a matching send call that delivers the exact block of log data to the TCP layer.

The logger does not wait for the log buffer to be full to flush it. A transaction commit will generate a flush request. If there is no request, loggw will still wake up from time to time to flush the log buffer. The size of each log flush is non-deterministic. When there are multiple client sessions, multiple requests can be combined. The logger is designed to be self tuning. If a flush takes a longer time, when the logger completes the flush, there are more outstanding requests, and therefore a stronger grouping effect on commit requests, improving performance by reducing the number of writes.

If there is only one client session, each commit will cause a log flush. If the synchronization mode is SYNC or NEARSYNC and the network is not very fast, the round trip messaging required by the synchronization mode can have a great impact on performance.

If the primary log buffer is large, each flush is likely to be large too. The send request to TCP can involve large blocks of data. The TCP layer should be tuned to handle such requests efficiently.

For HADR to work efficiently, a high speed, high capacity network between the primary and standby database is highly recommended. This will enable the standby to receive and acknowledge the logs as quickly as possible. We also recommend that the bandwidth of the network link be greater than the bandwidth of the logs generated. The network is vital to the performance of HADR, so it is important to tune the network. For better performance in an HADR environment, the network between both databases should be set up correctly. In terms of AIX TCP tuning, set these two OS parameters that affect the network performance:

► `tcp_recvspace`:

The `tcp_recvspace` tunable specifies how many bytes of data the receiving system can buffer in the kernel on the receiving sockets queue. The `tcp_recvspace` tunable is also used by the TCP protocol to set the TCP window size, which the TCP uses to limit how many bytes of data it will send to the receiver to ensure that the receiver has enough space to buffer the data. The `tcp_recvspace` tunable is a key parameter for TCP performance because TCP must be able to transmit multiple packets into the network to ensure that the network pipeline is full. If TCP cannot keep enough packets in the pipeline, then performance suffers. You can set the `tcp_recvspace` tunable by `no -o tcp_recvspace=[value]` command. You can also set interface-specific values of `tcp_recvspace` from `smit chinnet` menu.

► `tcp_sendspace`:

The `tcp_sendspace` tunable specifies how many bytes of data the sending application can buffer in the kernel before the application is blocked on a send call. The TCP-socket send buffer is used to buffer the application data before it is sent to the receiver by the TCP protocol. The default size of the send buffer is basically specified by the `tcp_sendspace` tunable value. You should set the `tcp_sendspace` tunable value at least as large as the `tcp_recvspace` value, and for higher speed adapters, the `tcp_sendspace` value should be at least twice the size of the `tcp_recvspace` value.

For more details on TCP tuning parameters, see the following Web site:

[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/tcp\\_streaming\\_workload\\_tuning.htm](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/tcp_streaming_workload_tuning.htm)

## Network delays

Network latency affects transaction performance in only SYNC and NEARSYNC modes. The slowdown in system performance as a result of using SYNC mode can be significantly larger than that of the other synchronization modes. In SYNC mode, the primary database sends log pages to the standby database only after the log pages have been successfully written to the primary database log disk.

In order to protect the integrity of the system, the primary database waits for an acknowledgement from the standby before notifying an application that a

transaction was prepared or committed. The standby database sends the acknowledgement only after it writes the received log pages to the standby database disk. The resulting overhead is the log write on the standby database plus round-trip messaging.

In NEARSYNC mode, the primary database writes and sends log pages in parallel. The primary then waits for an acknowledgement from the standby. The standby database acknowledges as soon as the log pages are received into its memory. On a fast network, the overhead to the primary database is minimal. The acknowledgement might have already arrived by the time the primary database finishes local log write.

For ASYNC mode, the log write and send are also in parallel; however, in this mode the primary database does not wait for an acknowledgement from the standby. Therefore, network delay is not an issue. Performance overhead is even smaller with ASYNC mode than with NEARSYNC mode.

## **Network down**

The primary database is continuously polling the network. If the network goes down, the primary database will detect the condition as soon as the primary host machine operating system detects the condition. If the primary machine and therefore the primary database can detect the failure, it will close the connection quickly. Closing the connection will cause the primary database to change from Peer state to disconnected state, unblocking pending transactions.

If the primary machine is unable to report network failure timely, DB2 will rely on HADR\_TIMEOUT to disconnect. If the primary does not receive any message from the standby for HADR\_TIMEOUT seconds, it will disconnect. In such a scenario, transactions on the primary are blocked while the primary waits on the timeout. In ASYNC mode, when the primary sends the logs, these could be buffered in the network path and delay the blocking of transactions on the primary. But as the primary keeps sending more logs, it will eventually cause congestion and the transactions are blocked.

When there is congestion, the standby knows that it cannot receive heartbeat from the primary anymore and updates its heartbeat receive time and continues to send heartbeat to the primary, so the standby will not drop connection after HADR\_TIMEOUT. On the primary side, because the primary is receiving heartbeat from the standby, the primary will not drop connection either, after the HADR\_TIMEOUT. Because the standby can send heartbeat and the primary can receive standby's heartbeat, the HADR pair will stay connected. In the worst case, an unresponsive network will cause primary transactions to be blocked until the primary does not receive any heartbeat or message from the standby for HADR\_TIMEOUT seconds. If no heartbeat is received by the primary from the standby until HADR\_TIMEOUT, the HADR state changes to DISCONNECTED.

When the primary state changes to DISCONNECTED, the primary gives up the transfer of the log records to the standby database. The primary database continues processing transactions even though it cannot transfer logs to the standby database. There is a possibility of data loss, if you execute a TAKEOVER BY FORCE command when there is a data gap between the primary database and the standby database. The primary will continue to listen on the HADR TCP/IP port waiting for the standby to come online. When the network recovers, the standby database tries to catch up to the primary until they return to peer status.

Figure 6-1 shows what happens when the network is down between the the primary and secondary.

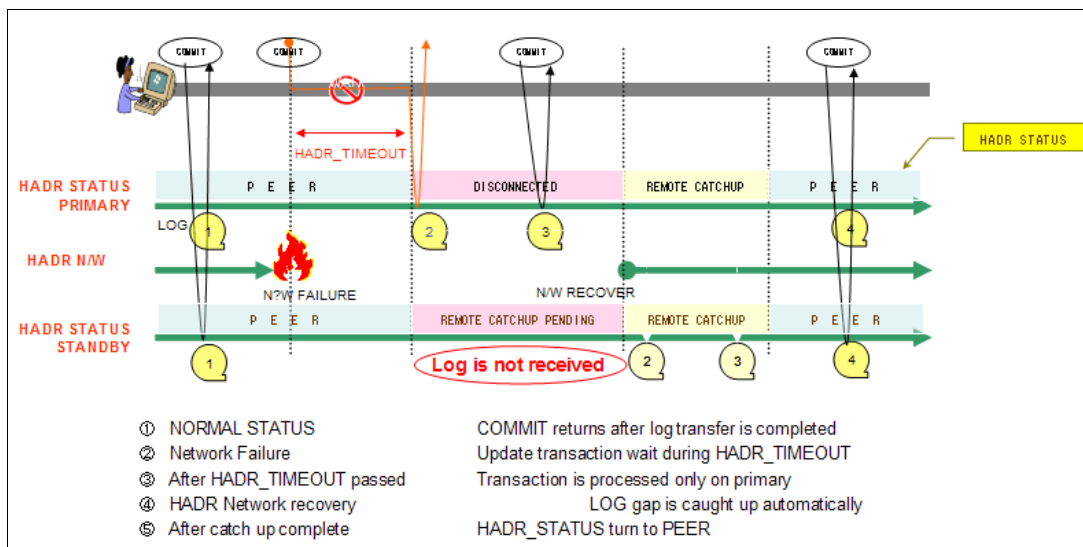


Figure 6-1 Behavior of HADR when the primary cannot transfer logs to the standby

## Network congestion

Network congestion can happen when the standby is behind in receiving and processing log data from the primary. This can cause the standby's log receive buffer to fill up, thereby preventing the buffer from receiving more log pages. This will cause a slowdown on the primary in all modes, including ASYNC mode.

In SYNC and NEARSYNC modes, if the primary database flushes its log buffer one more time, the data is likely to be buffered in the network pipeline, consisting of the primary machine, the network, and the standby database. Because the standby database does not have free log receive buffer (DB2\_HADR\_BUF\_SIZE) to receive it, it cannot acknowledge. So the primary is blocked, waiting for the acknowledgement. Congestion is not likely to occur for SYNC and NEARSYNC modes as unless this log data is acknowledged, and no more logs are sent by the blocked primary.

In ASYNC mode, the primary will continue to send logs until the network pipeline fills up. The network pipeline consists of the TCP buffers on the primary machine, the TCP buffers on the standby machine, the network path between the primary and the standby, and the HADR standby log receive buffer. When the primary cannot send any more logs, congestion can occur.

Large jobs such as table reorganization can flood the standby and cause congestion as the standby database is replaying log records that take a long time to replay. If the primary load has sharp peaks, they might be absorbed by a larger standby buffer.

You can mitigate the congestion problem by tuning the TCP/IP network to increase TCP/IP network buffering and also increase DB2\_HADR\_BUF\_SIZE registry variable to increase the log receive buffer on the standby. But a larger buffer would not help if the primary has a sustained log rate higher than what the standby can handle. Snapshot or db2pd will report connection status as “congested”. Congestion is reported by the `hadr_connect_status` monitor element.

### **6.3.7 Avoiding transaction loss in an HADR with HA cluster software**

An HADR with HA cluster software implementation automates the HADR takeover process. If a network error is detected, the primary database will enter the disconnected state and let transactions proceed. This behavior maximizes the availability of HADR database. Although the transactions can still be committed during the time the database is in a disconnected state, these transactions will not reach the standby and exist only on the primary database.

If the primary database goes down before the network is recovered, the automatic takeover process starts. Because the primary database is down, the standby will take over the primary by force. The log gaps are not handled and the log records that were not transferred to the standby database, yet are lost on the new primary database. Figure 6-2 shows the behavior of an HADR database when the primary database cannot transfer logs to the standby database.



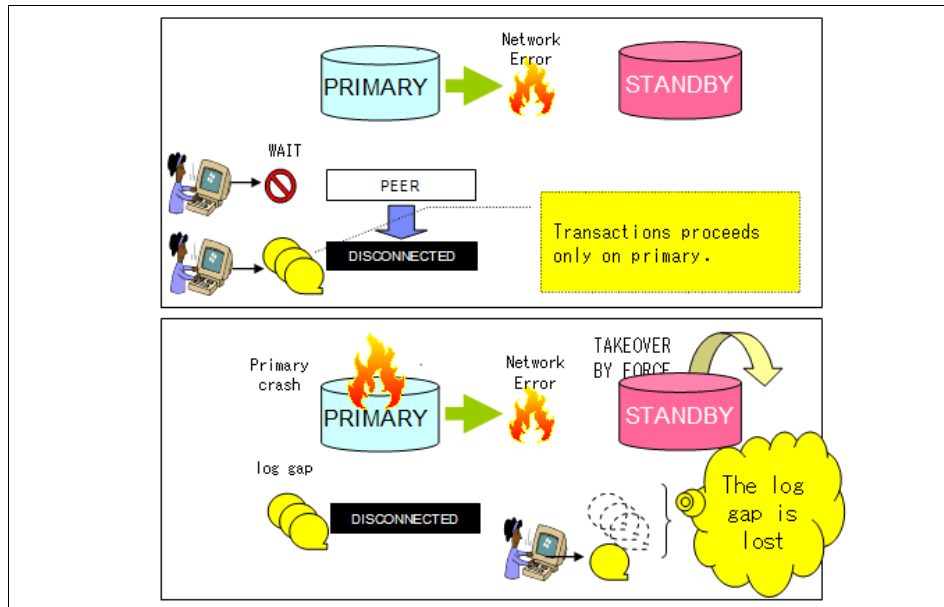


Figure 6-2 HADR behavior when the primary cannot transfer logs to the standby

**Note:** To reduce the possibility of network error, we recommend that you have duplicate networks for HADR communication.

To avoid data loss, the automated failover should only be performed if the HADR is in Peer state when the primary crashed. Before you execute HADR takeover, ensure that the HADR is in Peer state, which means that there is no log gap between databases in SYNC or NSYNC mode. ASYNC mode is not discussed here, because this mode does not guarantee the integrity of databases by its characteristics.

By monitoring the HADR status properly and adding a handling logic in an HACMP clustered environment, we can guarantee that the committed data will not be lost after the standby system took over the database. Figure 6-3 illustrates the concept of saving the committed data after the standby takeover.

The steps are as follows:

1. Monitor diag.log on the primary database.
2. Notify the HADR status.
3. Add integrity check logic to the takeover script.

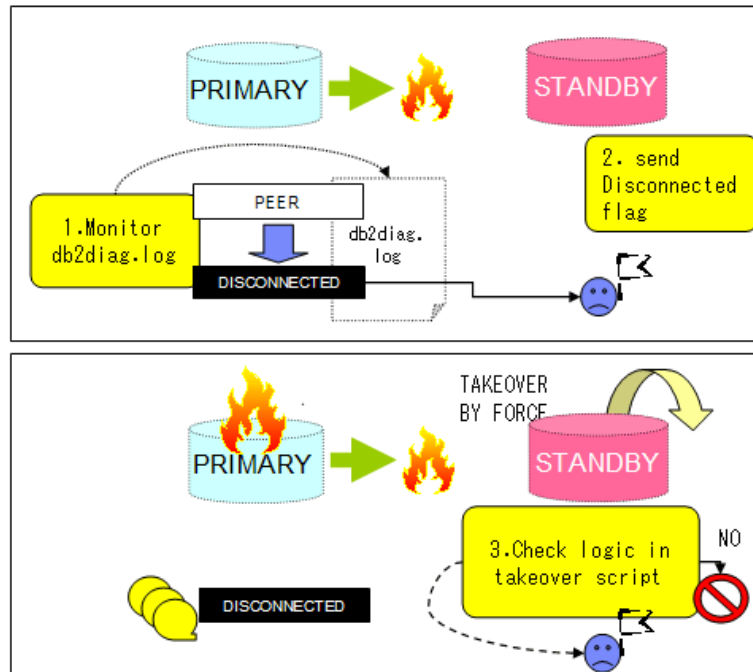


Figure 6-3 Avoiding data loss in an HADR with HA cluster software

1. Monitor diag.log on the primary database:

You have to detect the change of HADR status immediately when it turns into disconnected status. The best way to do this is by monitoring db2diag.log messages on the primary database. Example 6-1 shows a message in db2diag.log when state changed from peer to RemoteCatchupPending, which means that the primary database is disconnected from the HADR pair and can accept new transactions, leaving the standby database behind.

Example 6-1 HADR status changed from Peer to RemoteCatchupPending

---

```

2006-10-25-00.32.28.487034+540 E179519490C354    LEVEL: Event
PID      : 8056                    TID   : 1          PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst                NODE  : 000        DB    : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-RemoteCatchupPending (was P-Peer)

```

---

Example 6-2 shows a message in db2diag.log when status changed from RemoteCatchupPending to Peer.

*Example 6-2 HADR status changed from RemoteCatchupPending to Peer*

---

```
2006-10-25-00.35.47.823847+540 E179520902C353    LEVEL: Event
PID      : 8056                    TID   : 1          PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst                NODE  : 000        DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-NearlyPeer (was P-RemoteCatchup)

2006-10-25-00.35.47.830199+540 E179521256C344    LEVEL: Event
PID      : 8056                    TID   : 1          PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst                NODE  : 000        DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-Peer (was P-NearlyPeer)
```

---

To keep tracking on db2diag.log, you can use the `tail` command. See Example 6-3 for a sample script (`tail_hadr_status.ksh`). When this monitoring script detects the changes of HADR status by tracking the messages in the db2diag.log, it writes the status to the file. For example, set “1” when HADR status is changed from peer to non-peer, and set “0” when HADR status is returned from non-peer to peer. This status flag file allows the standby system to know the HADR state when the primary is down.

*Example 6-3 tail\_hadr\_status.ksh*

---

```
tail -0 -f $DIAGLOG | awk '
/CHANGE \: HADR state set to P-RemoteCatchupPending \(was
P-Peer\) / {system("'"$WRITE_STATUS"' 1")}
/CHANGE \: HADR state set to P-Peer \(was P-NearlyPeer\) /
{system("'"$WRITE_STATUS"' 0')}
```

---

To bring this monitoring script up and running at any time, run the script with the `nohup` option on the primary node:

```
#nohup tail_hadr_status.ksh &
```

We provide a complete HADR monitoring script in A.3, “`hadr_monitor.ksh`” on page 815.

## 2. Notify the HADR status:

Because the takeover command is issued on the standby database, the standby system has to know the HADR status of the primary database to take proper actions. In other words, this status flag should be accessible from the standby node even though the primary node is crashed. In our example, the status flag is on the standby node and updated from the primary node using a remote command. Especially in an HACMP cluster, `c1_nodcmd` is useful because this command can use any available network defined in HACMP cluster, which means `c1_nodcmd` can be executed over one of the available network even if a network interface for HADR is down. This is because HACMP is configured with multiple network cards in many cases. Refer to Chapter 8, “DB2 with TSA” on page 233 for more details.

Example 6-4 shows a code snippet of the update flag file script (`write_status.ksh`). This script is included in the `db2diag.log` monitoring script `tail_hadr_status.ksh` as a function (`write_status.ksh`). Parameter value `$1` (0 or 1) is given by `tail_hadr_status.ksh`, and `c1_nodcmd` writes it in the flag file on the standby node.

*Example 6-4* `write_status.ksh`

---

.....

```
usr/es/sbin/cluster/sbin/c1_nodcmd -cspoc '-n ${REMOTENODE}' "echo $1 > $STATUS_FLAG"
```

.....

---

## 3. Add integrity check logic to the takeover script:

In the takeover script, you need to add the logic to check the status flag before executing the TAKEOVER HADR command. It should failover only if the primary database had been in Peer state before it crashed. If the primary database was not in Peer state before the moment it crashed, it will not execute takeover automatically and only notify that user intervention is required to bring the database online. The user could recover the primary server from the crash, or copy all logs from the primary node to the standby node, run local catchup, and then takeover on the standby database.

Figure 6-4 shows a summary of the events occurred and the actions taken to prevent data loss:

1. HADR Network failure occurs.
2. HADR\_STATUS turns from peer to disconnected. The monitoring process catches the message in `db2diag.log` on the primary database and sends the flag file to the standby node.

3. Transactions are proceeded only on the primary database, not transferred to the standby.
4. The HADR network recovers from the error and log catch up is automatically run.
5. After completing catch up, HADR status returns to Peer state. The monitor script catches the message in db2diag.log on the primary database and sends the flag file to the standby node notifying that the HADR status had been back to peer status.
6. Meanwhile the flag is set to non-peer status "1" in event and kept the same status till event 5. We do not want to execute takeover by force because there is the possibility of data loss. We can add the logic in the takeover scripts to check this flag before issuing the TAKEOVER HADR command with the FORCE option.

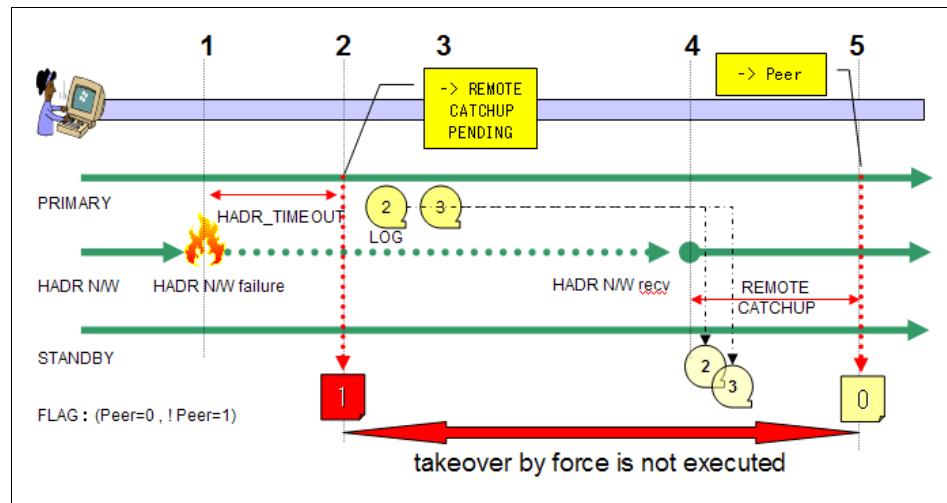


Figure 6-4 The behavior of the implementation

### 6.3.8 Avoiding transaction loss by using the peer window

The HADR cluster without the peer window has a small possibility to lose the updated data. To avoid this issue, we have to adopt the carefully designed architecture that is described in 6.3.7, "Avoiding transaction loss in an HADR with HA cluster software" on page 158. Now, on DB2 9.5, we can use the peer window to avoid this issue. In this section, we discuss the peer window and show the typical scenarios of HADR takeover without the risk of the data loss.

## DisconnectedPeer state

The HADR peer window introduces the new HADR state: DisconnectedPeer. If the HADR peer window is enabled, the HADR state will not be changed from the peer state to the RemoteCatchPending state directly. Instead, the HADR state is changed to DisconnectedPeer first. We call this period the *peer window phase*. In this phase, the primary database does not accept the commit requests. This means that the standby database can be switched to the primary database without the risk of data loss in this peer window phase.

Figure 6-5 illustrates what happens during the DisconnectedPeer state:

1. The HADR pair can communicate each other in normal operation.
2. During the peer window phase, all the commit requests are suspended because the primary database waits for the acknowledgement from the standby database.

If HADR\_PEER\_WINDOW is not set, the primary database accepts the commit request after the expiration of HADR\_TIMEOUT.

3. During the DisconnectedPeer state, the primary database cannot accept the commit requests. Therefore we can perform HADR takeover without the data loss.

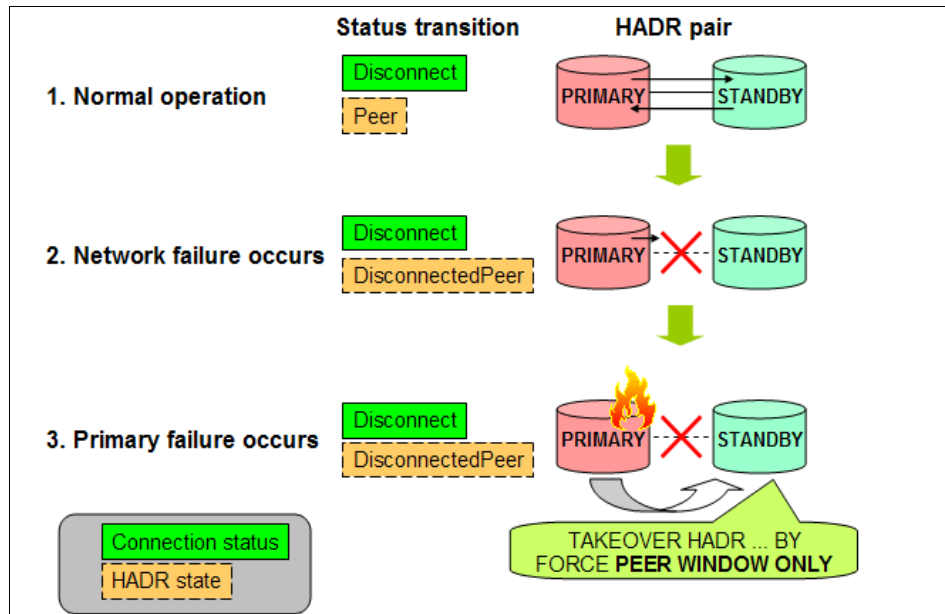


Figure 6-5 The typical behavior of the HADR with peer window

## Takeover of HADR without the risk of data loss

To perform the takeover without data loss, we have to assure that the takeover is taking place only when there is no unsent data in the primary database. We use the “PEER WINDOW ONLY” keyword to achieve this. The **TAKEOVER HADR** command with the **PEER WINDOW ONLY** keyword only succeeds when the HADR pair can take over without the risk of data loss. The following two scenarios illustrate how the peer window works when performing the HADR takeover.

### ***Scenario 1: A successful takeover in the peer window phase***

Figure 6-6 illustrates the progress of a successful takeover within the peer window phase:

1. The HADR pair is in the normal operation and HADR state is Peer. During the Peer state, the primary database sends the heartbeat with the PeerWindowEnd timestamp to the standby database regularly. In this example, the standby database receives the last heartbeat at 10:00:00 and recognizes that the PeerWindowEnd timestamp is 10:01:00.
2. A network failure occurred at 10:00:05.
3. The HADR state of the primary database changes to DisconnectedPeer (not RemoteCatchupPending) after the expiration of HADR\_TIMEOUT. The DisconnectedPeer state is kept until the time specified in the HADR\_PEER\_WINDOW db cfg parameter is up.
4. The primary database fails during the peer window phase.
5. Start the takeover by the **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** command on the standby database.
6. Complete the takeover by the time of PeerWindowEnd without the unsent transaction log.
7. If the primary database did not fail, after the peer window phase on the primary database has expired, the primary database starts to accept the update transactions from this point.

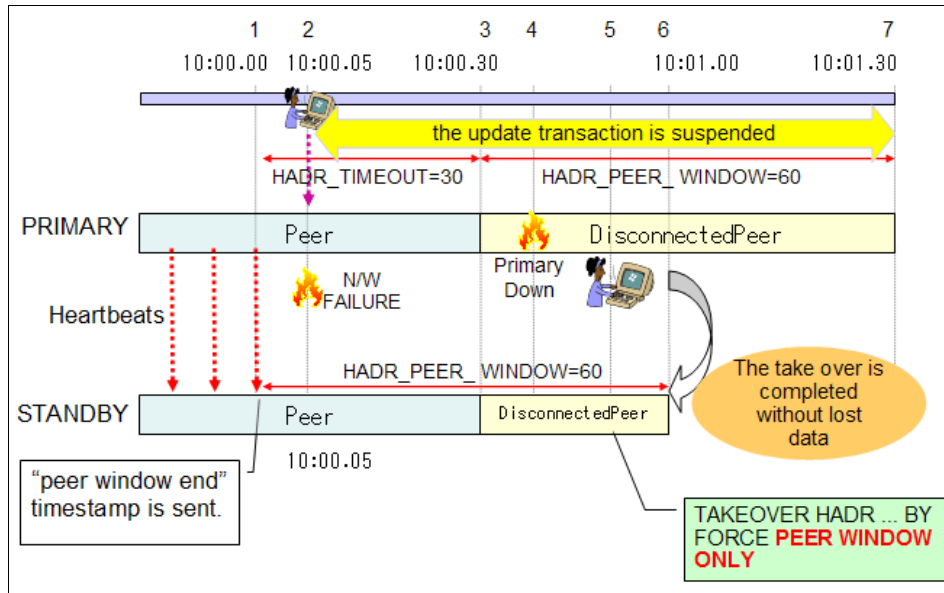


Figure 6-6 The successful takeover in the peer window phase

### Scenario 2: A failed takeover outside of the peer window phase

Figure 6-7 illustrates a failed takeover scenario due to the expiration of the peer window phase.

1. The HADR pair is in the normal operation and HADR state is Peer. During the Peer state, the primary database sends the heartbeat with the PeerWindowEnd timestamp to the standby database regularly. In this example, the standby database receives the last heartbeat at 10:00:00 and the standby database recognizes that the PeerWindowEnd timestamp is 10:01:00.
2. A network failure occurred on the HADR communication network at 10:00:05.
3. The HADR state of the primary database changes to DisconnectedPeer (not RemoteCatchupPending) after the expiration of HADR\_TIMEOUT. The DisconnectedPeer state is kept until the time specified in the HADR\_PEER\_WINDOW db cfg parameter is up.
4. The peer window phase expires on the standby database.
5. The primary database start to accept the update transactions because the peer window phase on the primary database expire at this point.
6. The primary database fails.



7. Start the takeover by **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** on the standby database. The **TAKEOVER HADR** command fails because the peer window phase has already expired.

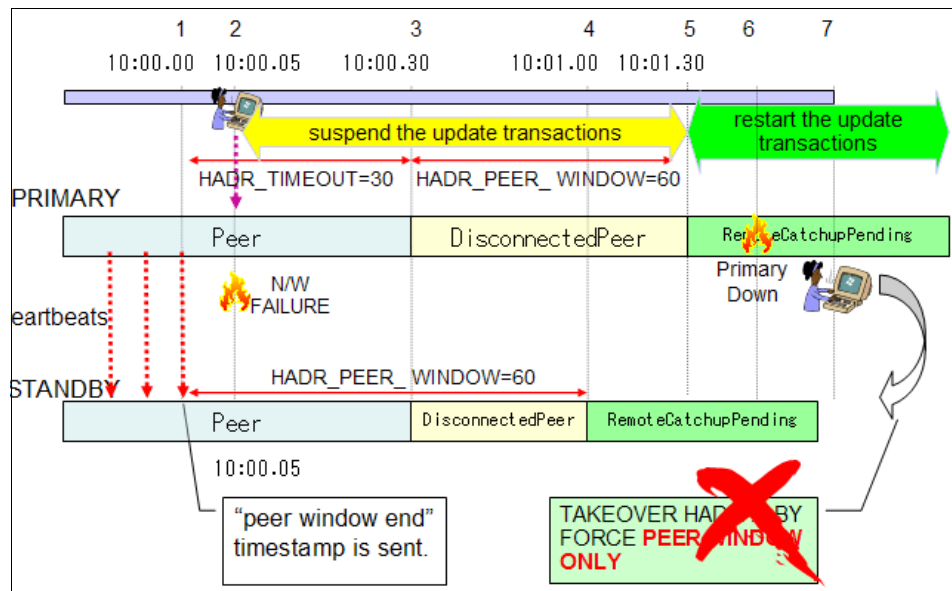


Figure 6-7 The failed takeover the outside of the peer window phase

## Considerations

There are a few considerations regarding the use of the peer window:

- ▶ Both servers should have the same system clock.

A PeerWindowEnd timestamp is generated on the primary database based on the system clock of the primary server. If the standby server has a different system clock, unexpected results might occur during the takeover. For example, if the system clock of the primary server is faster than the system clock of the standby server, the peer window phase on the standby database can continue longer than the primary database. This can cause the standby database to take over the primary database with unsent transaction data even if the **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** command is used.

- ▶ You must incorporate the time of the peer window into the application timeouts.

In general, the application timeouts should be longer than all the related timeout value of the database configuration parameters, otherwise the application can be timeout faster than database operations. Therefore, when you use the peer window, you must incorporate the peer window time into the

application timeouts. When the primary database cannot send the transaction logs to the standby database, the update transactions are suspended at least the time of HADR\_PEER\_WINDOW. Moreover, if the failure is caused by the network environment, the primary database waits to process the update transactions up to the total time of HADR\_TIMEOUT and HADR\_PEER\_WINDOW.

- ▶ If the HADR is with other cluster software, the time of HADR\_PEER\_WINDOW must be longer than the takeover time of the cluster.

When the HADR takeover is performed by the cluster software, your cluster script should use the **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** command. The takeover HADR command with **PEER WINDOW ONLY** requires the DisconnectedPeer status. If the HADR status has already been changed to RemoteCatchupPending before the execution of the takeover command, this script will fail with the message SQL1770N.

- ▶ If you use db2haicu for configuring the HADR cluster, the value of HADR\_PEER\_WINDOW must be larger than 120.

This is one of the prerequisites for db2haicu. For more information about db2haicu, refer Chapter 12, “Configuring clusters using the DB2 9.5 High Availability Feature” on page 477.

- ▶ If the HADR\_SYNCMODE is set to the ASYNC mode, DB2 ignores The value of HADR\_PEER\_WINDOW.

In the ASYNC mode, the primary database does not wait for the acknowledgement from the standby database. DB2 assumes that the value of HADR\_PEER\_WINDOW is “0”.

### 6.3.9 Index logging

By default, the index build creates a log “create object” record, but index pages within the object are not logged. The index is then marked as “pending rebuild” and is rebuilt after recovery is complete. This is undesirable for HADR, because a standby has pending build indexes that are rebuilt when someone accesses them. On each takeover you will have indexes marked as bad, and will need an index rebuild. Index rebuild can take a long time after failover.

The LOGINDEXBUILD database configuration parameter must be set to ON to establish a rule at the database level to log all indexes. This controls whether to log index build or not. Index update is always logged. This parameter applies at index creation, recreation, and table reorganization. The default is OFF. We recommend setting this parameter to ON for HADR databases.

There is a table level log index attribute that overrides the database level configuration parameter. This table level parameter is set with the ALTER TABLE statement LOG INDEX BUILD option, which can be set to the following values:

- ▶ ON: Logs index build
- ▶ OFF: Does not log index build
- ▶ NULL: Defaults to what LOGINDEXBUILD is set in database configuration

There might be an impact to transaction performance if LOGINDEXBUILD is ON. The impact on performance depends on the amount logs generated and number of indexes rebuilt or reorged, the HADR synchronization mode, and the network availability and the standby's ability to keep up with the primary. When the LOGINDEXBUILD is ON, issuing a REORG command on large number of tables that also involves reorging indexes could impact transaction performance.

The database manager and database level configuration parameter INDEXREC specifies when the invalid index is to be re-built. Refer to “DB2 HADR configuration parameters” on page 144 for more details on the values for this parameter.

### 6.3.10 Read on the standby

HADR has certain restrictions; the standby database is not a connectable database. You cannot update the database from the standby, nor can you issue any read-only queries on the standby. A fully accessible standby database would provide businesses with an opportunity for data mining without impacting users' access on the primary database.

If there is a customer requirement to read from a standby, then the workaround is to use VERITAS Storage Foundation for DB2 HADR. For more details on this implementation, read the white paper, *Accessible Standby Database in IBM DB2 UDB HADR Environment Using VERITAS Volume Instant Snapshots*, at the Web site:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/vrts-hadr.pdf>

This article describes the design and implementation of a solution for providing a fully accessible point-in-time copy of the standby database in a DB2 UDB HADR environment. Although this copy of the standby database is fully accessible, it is separated from the HADR pair, so updates made to it are not automatically applied to the main (HADR) pair of database copies.

This solution leverages a new feature called *instant space optimized volume snapshot* by which a single point-in-time copy of the storage can be created with only a small amount of additional disk space. Only storage for changed blocks is required. This article discusses how to activate a point-in-time copy of the standby database without impacting the active HADR databases. The space optimized volume snapshot feature enables the capability to produce a clean copy of the standby database without a WRITE SUSPEND on the primary database during the snapshot procedure.

Figure 6-8 illustrates the Veritas Storage foundation for DB2.

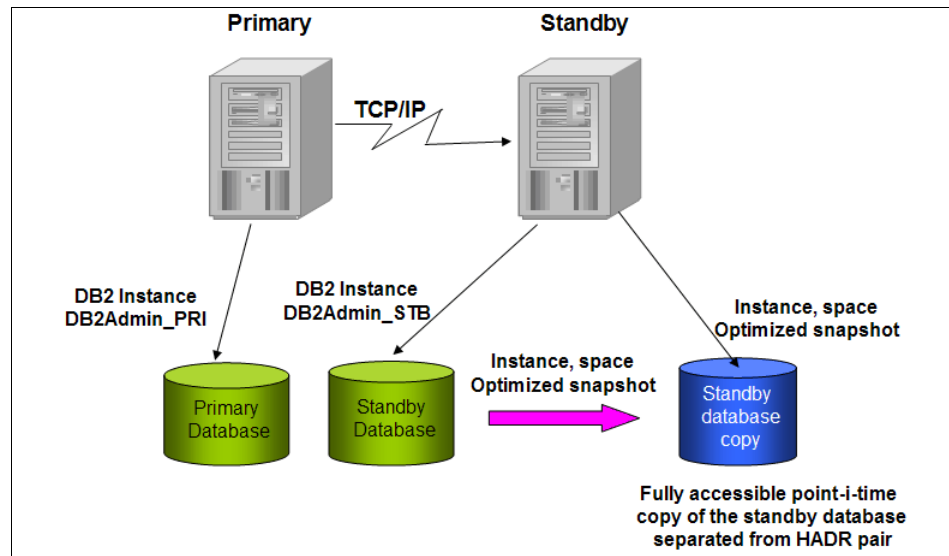


Figure 6-8 Veritas Storage foundation for DB2 overview

### 6.3.11 Backup from standby image with Flash Copy

With the Storage Copy function, you can utilize the ability to back up the HADR standby database without impacting the performance of the primary database.

Here we outline the steps to take a backup image from the standby database:

1. Database backup on the standby server:

Suppose you have another storage area for the snapshot of the standby database, and FlashCopy is configured between the storage area for the standby database and the snapshot. Do the following steps:

- a. Deactivate the database on the standby database.
- b. Stop the database instance.
- c. Unmount all the file systems and varyoff volume groups. To ensure the consistency of the snapshot image, no write activity is allowed during FlashCopy logical copy. So it is preferable to unmount file systems (and varyoff volume groups).
- d. Use a Storage Copy function such as Flash Copy to take a snapshot of the storage image from the standby database, which includes the database directory and table space containers. Basically, the logical copy completes in a moment before waiting for the physical copy of whole storage area. With IBM Total Data Storage, after the logical copy is finished, you can activate standby image again.
- e. Activate (varyon) volume groups, mount file systems, and activate the database. Note that when the standby database is deactivated, you might not have the latest standby database.

2. Restoring to the primary server:

If you have to restore the database from this backup to the primary server, you must do the following steps:

- a. Restore all the DB2 related files to the correct original location on the primary server from this backup image. The restored database should be in database rollforward pending state already.
- b. Make all the log files available (including the active log files and archived log files from the primary server) to be processed.
- c. When the backup image was made on the standby server, its HADR status was on standby mode. Also, the HADR configuration parameters are copied from the standby database. They should be updated for the primary database.
- d. Apply the log files from the original primary database to the restored database.

- e. Restart the HADR (assuming that the standby database is still up and running).
3. Restoring to the standby server:

If the standby database becomes unusable due to a hardware problem, the split backup can be restored to the original standby database. No special configuration is required. After the standby database files have been restored, start HADR as standby. The standby database should now go into catchup mode, with the primary database server retrieving the log files and shipping the log records over to the standby database until it reaches Peer state.

For more details, see Chapter 7 of the document listed at the following Web site:

<ftp://submit.boulder.ibm.com/sales/ssi/sa/st/e/0rwhr-6h3ppe/HADR-in-an-SAP-implementation-04Aug2005.PDF>

### 6.3.12 Replicating load data

When LOAD utility is used to move data into a table, the data is not included in the log file. There are some special requirements to replicate load data in the HADR environment. The basic rule is that the load operation is replicated only if there is a copy of the loaded data available for the standby to obtain the data.

The copy device must be available when the standby replays the load operation. The standby might attempt to access the copy any time after the primary completes the load. If a load operation is executed on the primary database with the NONRECOVERABLE option, the command will execute on the primary database and the table on the standby database is marked bad. The standby database will skip future log records that pertain to this table. You can choose to issue the LOAD command with the COPY YES and REPLACE options specified to bring the table back, or you can drop the table to recover the space.

For more details, see DB2 Information Center: **Administering** → **database systems** → **High Availability** → **Managing HADR** → **Database configurations for HADR**, and search for “Load operations and HADR”.

The following methods can be used for replicating load data to the standby database.

► Using NFS shared disk:

You can utilize network file system (NFS) to share the load copy image on both primary and standby nodes. See Figure 6-9. On the standby node, be sure to mount the NFS shared directory to the exactly same point as the primary can see.

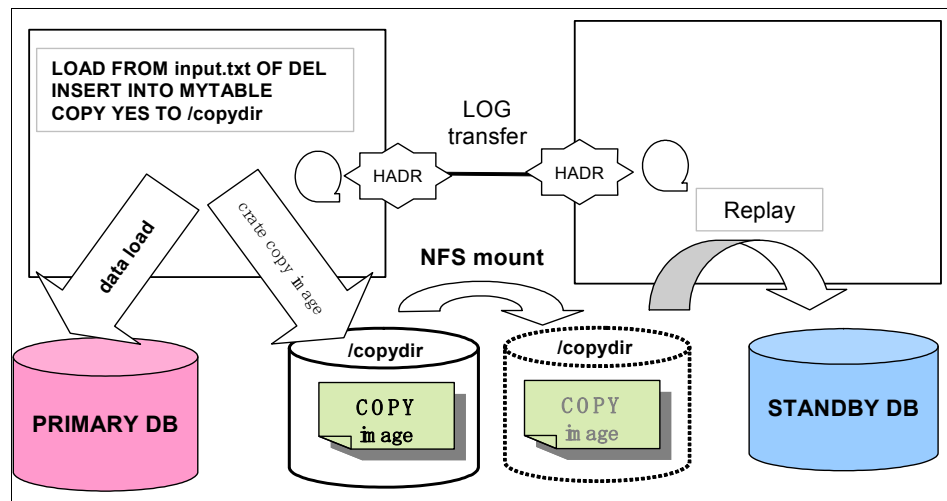


Figure 6-9 Using NFS to share data

2. Pause (Deactivate) the standby database during transferring load copy:

If you transfer the copy image by some means such as filecopy or physical tape media, we recommend that the standby be stopped (db2 deactivated db) before the primary runs the load. After the primary finishes the load and copy is in place on the standby, restart the standby (db2 activate db). It will then reconnect and replay the load.

Make sure that instance owner has proper access permission to the copy file. For example, in AIX system, the permissions of copy image is set to 640 (rw-, r--, ---), instance owner of the standby database should have same UID (User ID) or belong to the same GID (Group ID) so that the copy image file generated by the primary instance is also accessible from the standby instance. If it is not accessible from the standby instance, the table space that includes the table with loaded data is marked as restore pending and cannot accept any more updates. You'll have to rebuild the standby database from the primary.

### 6.3.13 Log archive and HADR

Log archive only happens on the primary. If the primary and the standby use independent log archive devices, with takeover back and forth, some log files are archived on one device and others on the other device. Log is never archived on the standby database even though you set up USEREXIT, LOGARCHMETH1 or LOGARCHMETH2. The standby writes logs it receives from the primary into its local log files. To avoid filling up the log path, it recycles log files after a file is no longer needed for the purposes such as crash recovery and takeover readiness.

For more information, see DB2 Information Center: **Administering** → **database systems** → **High Availability** → **high availability disaster recovery overview** → **Standby database states**.

### 6.3.14 Database restore considerations

Following are some considerations for database restore:

- ▶ Redirected restore is not supported.

“Redirected” here means redirecting table space containers. Database directory (restore database ... to) and log directory (restore database ... newlogpath ...) changes are supported. Table space containers created by relative paths are restored to paths relative to the new database directory.

- ▶ Restoring the standby database has certain requirements.

When creating the standby database by restore from primary database, the restore command(s) should leave the database in rollforward pending state, otherwise, start HADR as standby will fail with HADR error SQL1767N start HADR cannot complete. Reason code = "1". Be sure *not* to specify the option, *without rollforward*, when you rebuild the standby database.





## DB2 and system upgrades

In this chapter we focus on applying DB2 fix packs with the no-downtime benefit that HADR brings, and DB2 version upgrades with minimized downtime using HADR databases. We also explore important considerations to keep in mind when performing changes to other system components that might require DB2 or the whole server to be recycled. Lastly, we provide an overview of what to keep in mind when updating certain database configuration parameters.

We cover the following topics separated into techniques for GUI and Command Line interfaces, so as to cater for both styles of DB2 administration on midrange platforms:

- ▶ DB2 fix pack rolling upgrades
- ▶ DB2 migration (version upgrade)
- ▶ Rolling OS/Application/DB2 configuration parameters

**Note:** To avoid confusion, we use the term “fix pack” in this chapter, which has been the standard IBM terminology for DB2 V9.1 and V9.5. The previous term for DB2 V7 and V8 was “FixPak”.

As of the time of writing, the URL for downloading DB2 fix packs for all available versions is:

<http://www-1.ibm.com/support/docview.wss?rs=71&uid=swg27007053>

In our text, we use the terms “primary” and “standby” interchangeably with “HADR Primary” and “HADR Standby”. We are referring in most cases to the initial role the DB2 HADR-paired database on each server plays in the normal production environment. We use the term “standard” in the context of a database which has either not had a **start hadr** command issued against it yet, or which has just had a **stop hadr** command issued against it. The values of either primary, standby, or standard are found in the output of the **get db cfg** command, in the “HADR database role” field, for example:

```
HADR database role = STANDARD
```

While we cover the Linux platform specifically in our examples, the commands and techniques are equally applicable to DB2 on all UNIX platforms.

## 7.1 DB2 fix pack rolling upgrades

HADR gives you high availability while applying DB2 fix packs through rolling upgrades. Your database down-time need only be literally momentary while you are switching roles between your database servers. With properly written applications using standard retry logic and handling of message SQL30108N if you are using Automatic Client Reroute (ACR), this effectively means no loss of data and no visible downtime to clients.

In this section we provide examples of a rolling upgrade fix pack apply on a pair of Microsoft Windows servers, to illustrate the procedure through Graphical User Interface (GUI) tools. We also give you an example of a rolling upgrade performed on a pair of Linux servers using the DB2 Command Line Processor (CLP).

These are the general steps to perform a rolling upgrade, where the HADR Primary database is never taken offline from users:

1. Check your system's HADR status.
2. Apply fix pack on the standby system.
3. Switch HADR roles from the standby system.
4. Apply the fix pack in the old primary system.
5. Switch HADR roles back to the original primary.

### 7.1.1 Rolling upgrade on Microsoft Windows GUI

In this section, we describe the procedure to apply a DB2 fix pack with no downtime on Microsoft Windows, from DB2 ESE V9.1 fix pack 1 to V9.1 fix pack 4a, using the DB2 Control Center Graphical User Interface (GUI) wherever practical. Figure 7-1 illustrates the rolling upgrade performed in our test lab environment.

We use two servers, PUGET and FAROE. Both have the database SAMPLE, as HADR Primary on PUGET, and as HADR Standby on FAROE. Our DB2 instance name is the default "DB2". The DB2 instance user id is "db2inst1", and is connected to the Local Administrator Group for Windows, in order to perform DB2 product installation tasks.

Figure 7-1 illustrates our rolling upgrade step sequence.

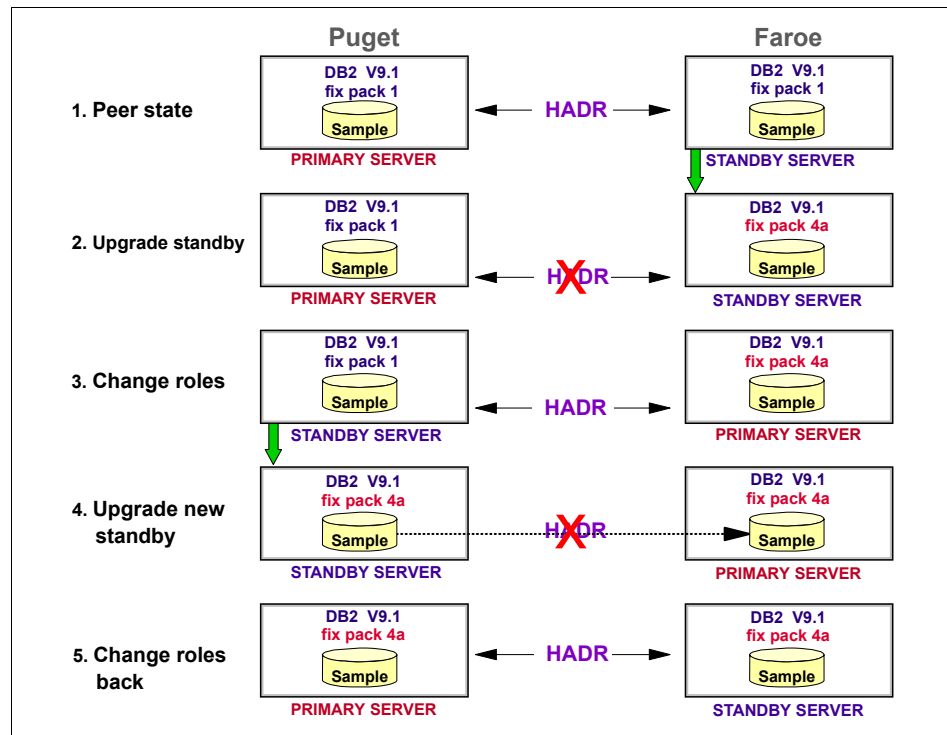


Figure 7-1 Rolling upgrade

The steps to apply the fix pack are as follows:

1. Extract the DB2 fix pack into a source location, on both servers. We do this to minimize the time that the HADR Standby is down, and to confirm the integrity of the fix pack download. We extract our fix pack to `c:\temp\v9fp4a` on both servers, as shown in Figure 7-2:

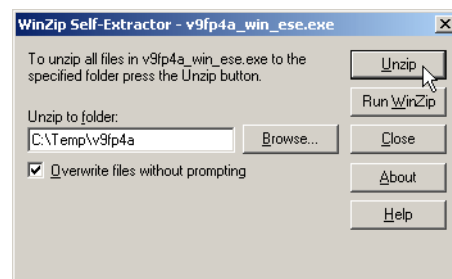


Figure 7-2 Extracting DB2 fix pack to an installation source directory

2. Confirm prerequisite status for Microsoft Windows.

First, your DB2 instance user ID must be a member of the local Administrators group. Confirm this with the local user manager Snap-In: **Start** → **Run**, type **lusrmgr.msc**, press Enter, shown in Figure 7-3.

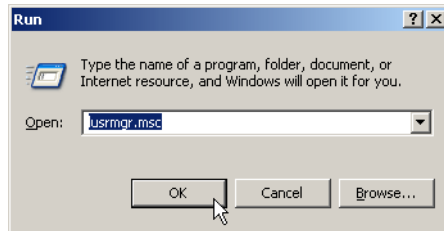


Figure 7-3 *lusrmgr.msc* snap-in to manage local users and groups

Figure 7-4 shows both our DB2 instance user and the DB2 Admin Server user in the Windows Local Administrators group.

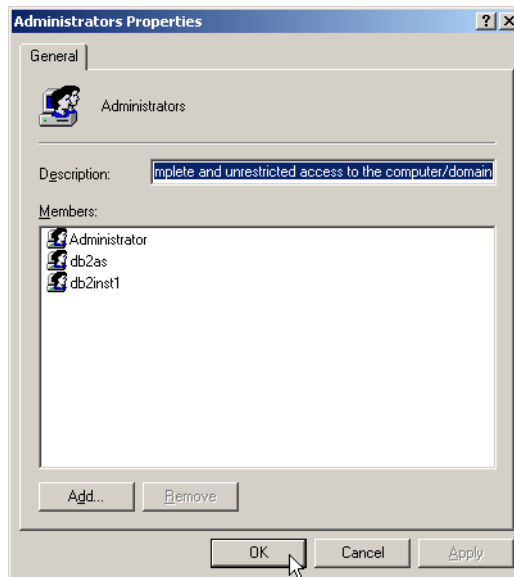


Figure 7-4 *Local user and group management - DB2 users in Administrators group*

Second, the HADR database pair should be in Connected, Peer state. Current HADR state can be most easily checked with the **db2pd -d <dbname> -hadr** command, or the Manage HADR window in the DB2 Control Center. See Chapter 4, “HADR administration and monitoring” on page 99 for the details of using these utilities.

To open a Manage HADR window from the Control Center, expand the object tree down to the database object, and right-click the database name. Select **High Availability Disaster Recovery** → **Manage ...**, and the output should resemble Figure 7-5.

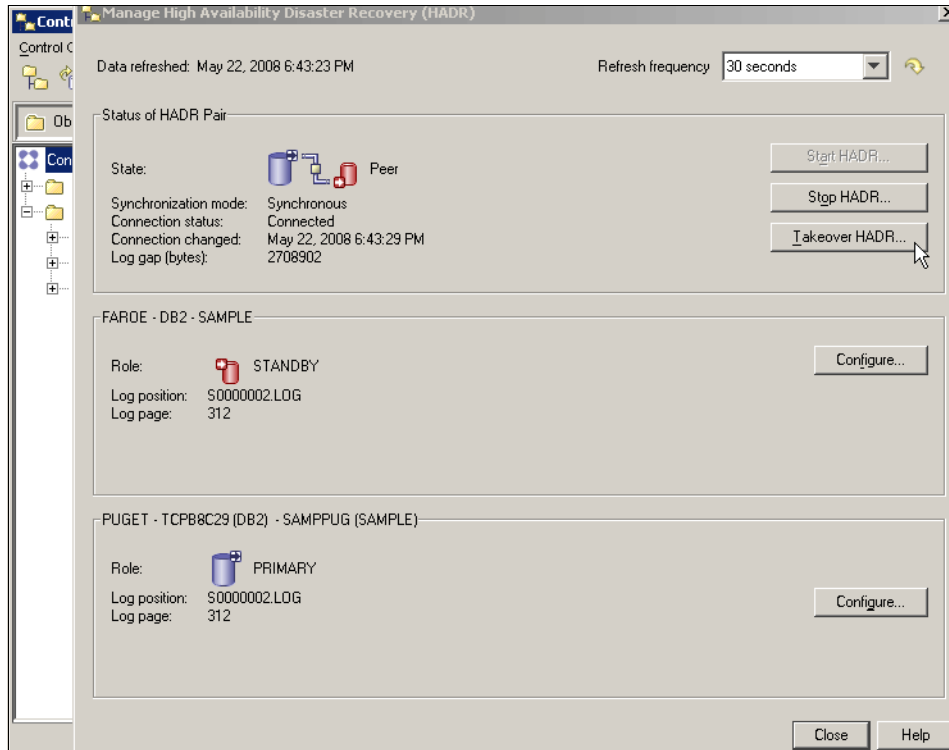


Figure 7-5 HADR in Connected, Peer state, ready for fix pack apply

3. Apply the DB2 fix pack code on the standby server (FAROE in our test case). Refer to the DB2 Information Center for more details on the actual fix pack apply:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/t0006352.htm>

Formerly, this information was contained in individual downloadable fix pack Readme text files, but it has been integrated into the DB2 Information Center as of DB2 V9.1.

Basic fix pack installation steps for DB2 ESE V9 fix pack 4a follow:

- a. Deactivate your standby database and stop the DB2 instance. This can be most easily achieved in one step. From the Control Center, expand the object tree down to the database object. Right-click the database name, and then click **Stop**. This will stop the instance as well as the database.

Figure 7-6 shows the resulting dialog where we click **OK** to confirm the stop action.

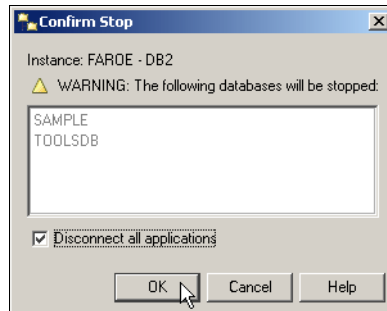


Figure 7-6 Confirm stop of database(s) and DB2 instance

- b. Stop all other DB2 prefixed services in the Microsoft Management Console Services Snap-in by:

**Start** → **Run**, type **services.msc**, and press Enter.

Figure 7-7 shows us in the process of stopping the DB2 Admin Server service, from the Services Snap-in.

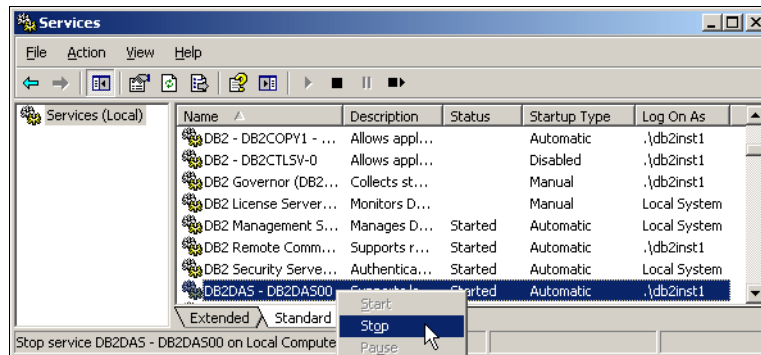


Figure 7-7 Stopping DB2 services

- c. Start Windows Explorer and open the folder where the extracted fix pack files are located. Our DB2 V9.1 ESE fp4a setup code is in:  
c:\temp\v9fp4a\ESE

To start the DB2 Setup Launchpad, double-click **setup.exe** from that folder. Select **Install a Product** from the list of options on the left, and the resulting dialog is shown in Figure 7-8.



Figure 7-8 Install a Product - Work with Existing

Click **Work with Existing**, in order to apply the fix pack to an existing DB2 copy location, rather than installing the new product code to a different copy location.



- d. Select the DB2 copy to work with.

If you only have a single DB2 copy location, it will already be highlighted as shown in as shown in Figure 7-9. Otherwise, choose the copy for your existing HADR pair. Click **Launch DB2 Setup wizard**.

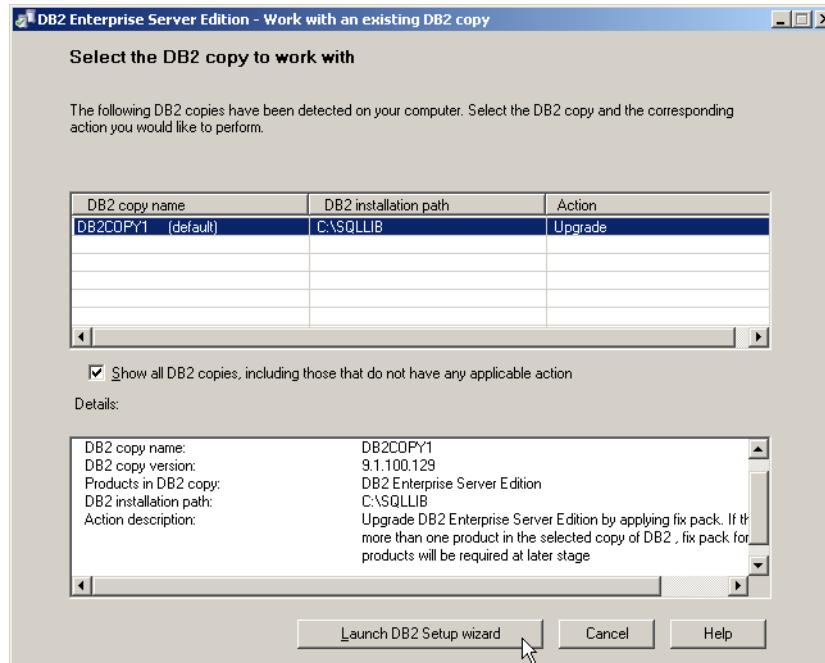


Figure 7-9 Select the DB2 copy to work with

- e. You might receive warning dialog boxes for processes still running, such as **db2sysstray.exe** in Figure 7-10. Click **Yes** for the wizard to stop them, or stop them manually and click **Yes**.



Figure 7-10 db2sysstray.exe was running in the background in the system tray

f. Installing DB2 Enterprise Server Edition - DB2COPY1.

A dialog with a progress bar appears as in Figure 7-11. Wait for it to complete.

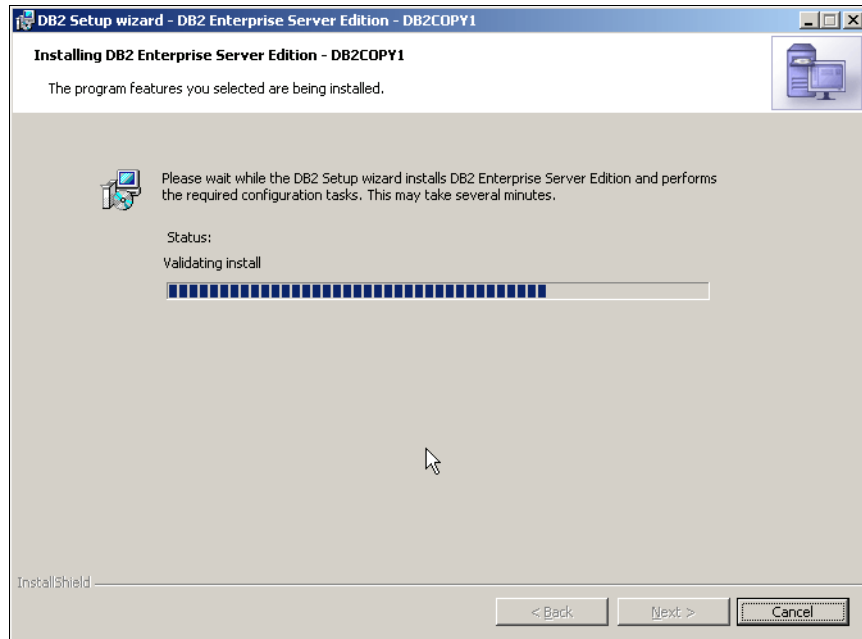


Figure 7-11 Progress of install

- g. The dialog shown in Figure 7-12 appears, notifying you that Setup is complete. Click **Finish**.

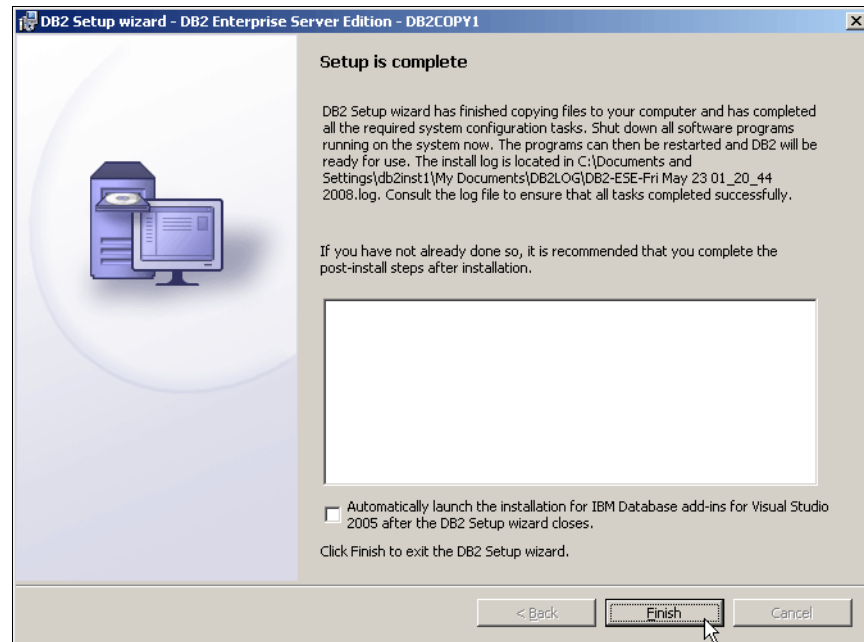


Figure 7-12 Setup is complete

- h. The dialog shown in Figure 7-13 appears, stating that Windows must be restarted.

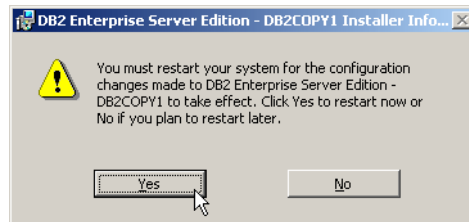


Figure 7-13 Windows restart required

Click **Yes** to immediately restart. After the restart, and logging back in as the DB2 instance user, the DB2 First Steps window appears, shown in Figure 7-14. Close it by clicking **Exit** in the left hand menu pane.

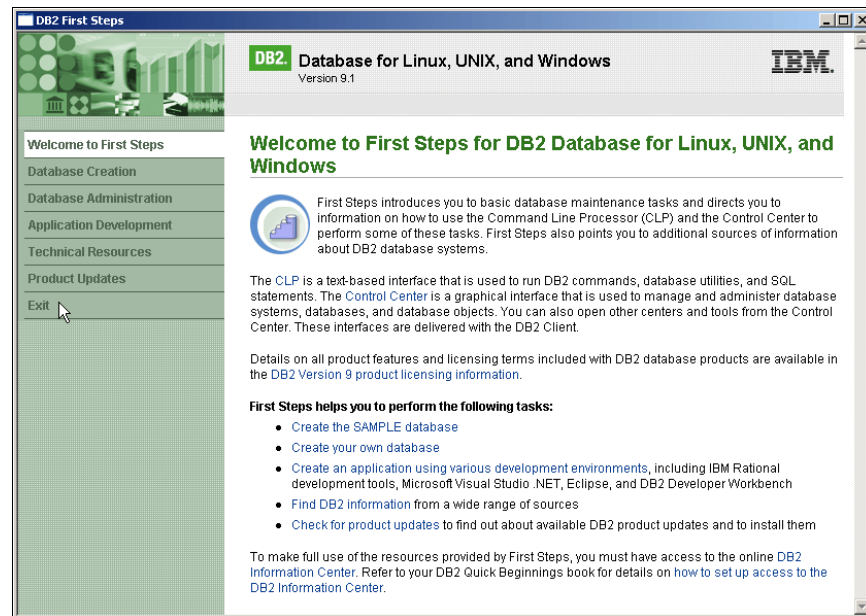


Figure 7-14 DB2 First Steps

- i. From the Microsoft Management Console Services Snap-in, confirm that DB2 services are running:

**Start** → **Run** → type **services.msc**, press Enter.

The following two processes should be started, as a minimum:

- DB2-DB2-0 (or whichever instance you are using for HADR)
- DB2DAS - DB2DAS00

**Note:** At this time you will not be able to issue the bind commands required as post-installation steps in the fix pack apply, because the database is in rollforward pending state. The binds are addressed later in the procedure, in step 5.b on page 191.

- j. Activate the HADR Standby database. This must be done through the CLP:

**Start** → **All Programs** → **IBM DB2** → **DB2COPY1 (default)** → **Command Line Tools** → **Command Window**

**db2 activate db sample**

- k. Confirm that HADR is in a Connected, Peer state.

Open the DB2 Control Center, expand the object tree down to the database object, right-click the database name. Select **High Availability Disaster Recovery** → **Manage ...**

Our Manage HADR dialog appears in Figure 7-15:

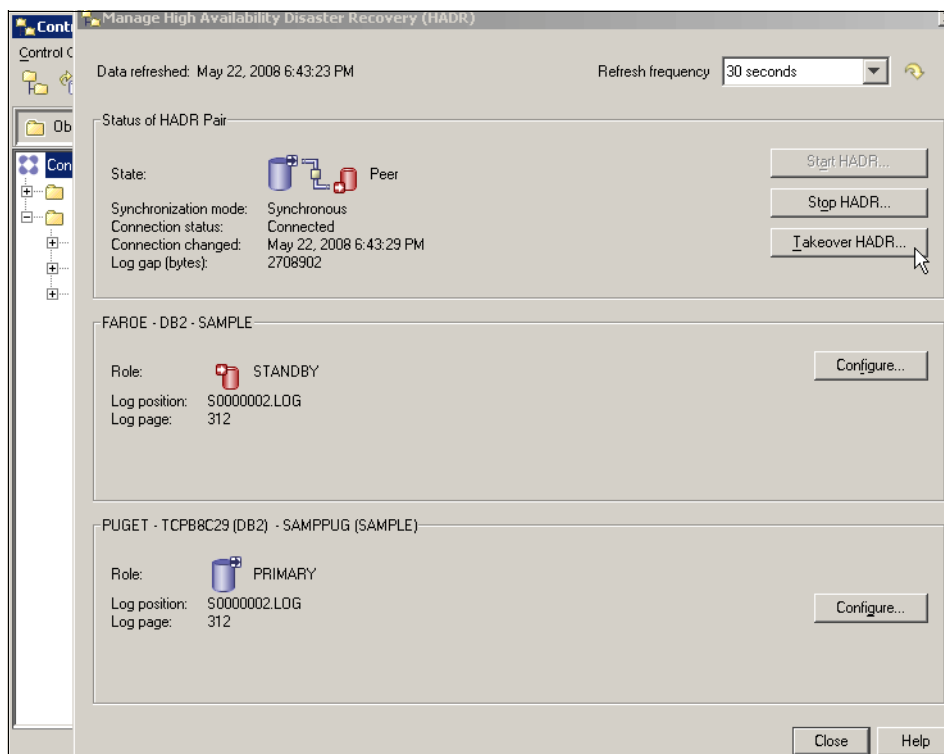


Figure 7-15 Manage HADR after first fix pack apply

4. Switch HADR roles.

In our example, server FAROE becomes the new HADR Primary and PUGET becomes HADR Standby.

- a. In the Manage High Availability Disaster Recovery (HADR) window, click **Takeover HADR ...** as shown in Figure 7-15.
- b. To switch roles (unforced takeover), select **Switch roles** in the Takeover HADR window shown in Figure 7-16, and click **OK**.

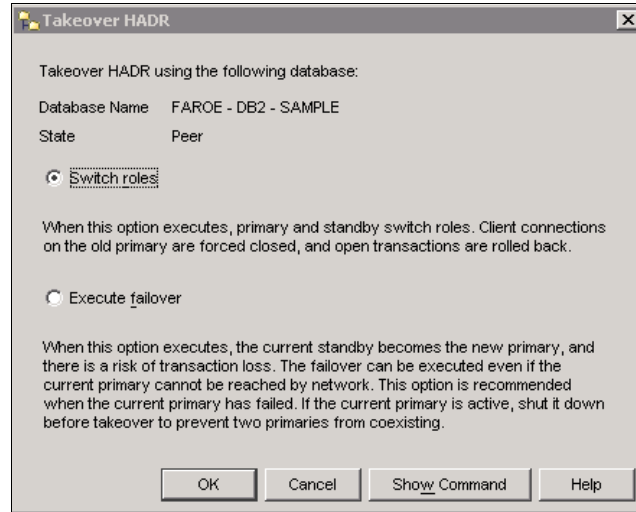
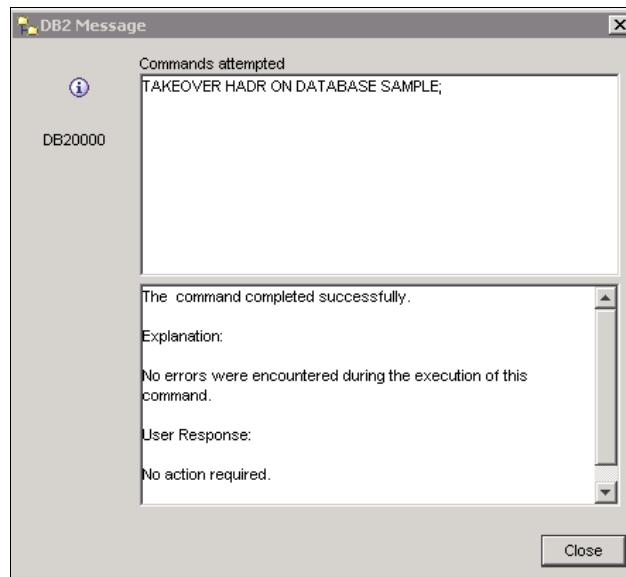


Figure 7-16 Switch HADR roles

- c. A DB2 message should come up to confirm success. See Figure 7-17. Click **Close**.



*Figure 7-17 Takeover successful message*

- d. The Manage High Availability Disaster Recovery (HADR) dialog should now show that the standby and primary roles have switched servers. In our example in Figure 7-18, FAROE is now the primary.

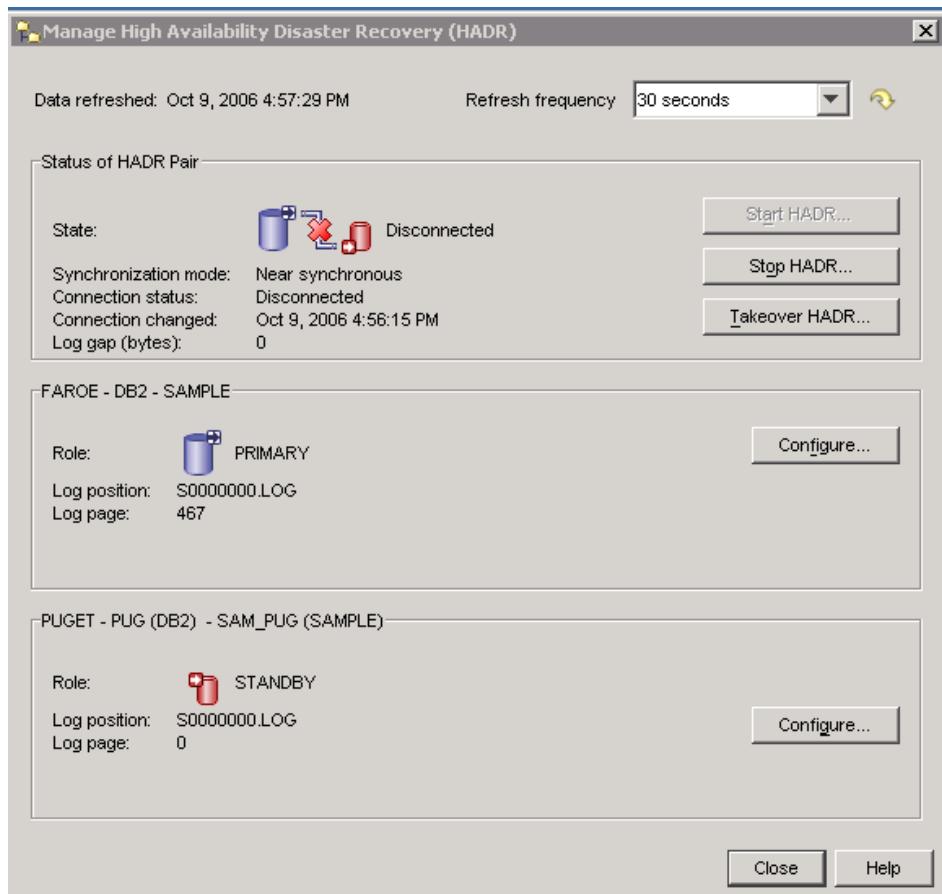


Figure 7-18 Successful takeover

Note that HADR supports the standby running a newer DB2 level than the primary, but not the other way around. Therefore, when the takeover is complete, HADR is in Disconnected state, and the new standby (old primary) database is deactivated, because it has an older DB2 level.

5. Apply the DB2 fix pack on the old HADR Primary server.
  - a. Repeat step number 3 on page 180, on the new standby server. In our example, we perform the fix pack apply on PUGET.



- b. Post-installation tasks such as the **db2updv9** command for Windows databases, and package binding for DB2 CLI cannot be performed from an HADR Standby database, so we have to connect to the current HADR Primary database - SAMPLE on FAROE in our case. When connected, execute the binds appropriate to your environment as described in the DB2 V9.1 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/c0025015.htm>

The binds and system catalog updates are logged, so HADR replays them on the standby to complete the fix pack apply.

- 6. (Optional) Switch roles back to the original primary if you want to restore your original server roles.
  - a. HADR status must be in Connected, Peer state to ensure all logs have been applied.
  - b. Repeat step number 3 to issue an unforced HADR takeover.

This concludes our rolling upgrade (fix pack apply) using the tools of a GUI environment.

## 7.1.2 Rolling upgrade on Linux Command Line

This section gives an example of a rolling fix pack apply in a command line environment. Our test case has two Linux servers, LEPUS as HADR Primary, and MENSA as HADR Standby, with database name SAMPLE. We perform a rolling upgrade from DB2 V8 fix pack 12 to fix pack 16, but this technique is applicable to any fix pack apply which is not a version or subversion upgrade or migration.

- 1. Check your system's HADR status.

HADR should be in Connected, Peer state. You can check your HADR status by **db2pd**, **get snapshot**, or the Manage HADR dialog if you have an X11 windowed interface working. We use **db2pd** as follows:

```
db2pd -d sample -hadr
```

We expect to see something similar to Example 7-2.

*Example 7-1 Output of db2pd showing HADR Connected, Peer state*

---

```
db2inst1@lepus:~/script> db2pd -d sample -hadr
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:21
```

```
HADR Information:
```

Role	State	SyncMode	HeartBeatsMissed	LogGapRunAvg (bytes)
Primary	<b>Peer</b>	Sync	0	0

ConnectStatus	ConnectTime	Timeout
<i>Connected</i>	Thu May 22 01:28:43 2008 (1211444923)	120
LocalHost	LocalService	
lepus	55101	
RemoteHost	RemoteService	RemoteInstance
mensa	55102	db2inst1
PrimaryFile	PrimaryPg	PrimaryLSN
S0000002.LOG	0	0x0000000001770000
StandByFile	StandByPg	StandByLSN
S0000000.LOG	0	0x0000000000FA0000

---

2. Apply fix pack on the standby server.
  - a. Deactivate the database on the standby server from the CLP. The standby is MENSA for our example:

```
db2 deactivate database sample
```
  - b. Stop DB2 from the CLP in preparation for the fix pack apply:

```
db2stop
```
  - c. Apply fix pack code. Refer to the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/t0006352.htm>
  - d. Perform instance update commands **db2iupdt** and **dasupdt** for UNIX platforms as described in the Information Center fix pack post-apply tasks:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/t0024995.htm>
  - e. Start DB2 from the CLP:

```
db2start
```
  - f. Activate the database on the standby from the CLP.

```
db2 activate database sample
```

**Note:** At this time you will not be able to issue the bind commands required in the fix pack apply, because the database will still be in rollforward pending. The binds are addressed in step 4b.

3. Switch roles from the standby system.
  - a. From the CLP on the standby database, issue the **takeover hadr** command. In our test case, from MENSA:  

```
db2 takeover hadr on database sample
```

After the take over, your HADR Connection status is *disconnected* because of the different levels of DB2.

Note that HADR supports the standby running a newer DB2 level than the primary, but not the other way around. Therefore, when the takeover is complete, the new standby database is deactivated, because it has an older DB2 level.
4. Apply the fix pack in the new standby server.
  - a. Repeat step number 2 on the new standby server. In our example, we perform the fix pack apply on LEPUS.
  - b. Post-installation tasks such as the **db2updv9** command for UNIX platform databases, and package binding for DB2 CLI cannot be performed from an HADR Standby database, so we have to connect to the current HADR Primary database - SAMPLE on MENSA in our case. When connected, execute the **db2updv9** command on each database on the HADR Primary, as well as the binds appropriate to your environment as described in the DB2 V9.1 Information Center:  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.uprun.doc/doc/t0024995.htm>  

The binds and system catalog table updates are logged, so HADR replays them on the standby to complete the fix pack apply.
5. (Optional) Switch HADR roles back to the original primary.
  - a. Your HADR status must be Peer to ensure that all logs have been applied.
  - b. Repeat step 3 to issue an unforced HADR takeover on the original primary server, currently new standby server, LEPUS in our test case.

This concludes our example of a rolling upgrade using the command line environment.

## 7.2 DB2 migration (version upgrade)

A rolling upgrade is not supported between DB2 version or major subversion migrations, such as version 8.2 to 9.1, or 9.1 to 9.5. For these events, plan for a database outage while the HADR primary database is updated. In this section we give an example of a DB2 migration on a Microsoft Windows operating system, using the DB2 Graphical User Interface (GUI) where practical, and on Linux, using the command line.

DB2 installations and upgrades have been significantly assisted since the general availability of DB2 V8 on Microsoft Windows platforms, and V9 on UNIX platforms, by containing the base code for an installation in the same downloadable package as each fix pack. This reduces downtime windows, and greatly simplifies installations and upgrades. Having everything in a single package also avoids many situations where post-installation steps are neglected or only performed before applying the fix pack rather than afterward, in a typical “Install the base code first then perform a fix pack apply on top” scenario.

For your own reference, documentation on related concepts involved in DB2 migrations can be found in the DB2 V9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.migration.doc/doc/c0023662.html>

The initial critical step of DB2 migrations is the **db2ckmig** command, which checks if databases are eligible for migration to the new DB2 version. New features of the **db2ckmig** command listing dependencies for success are covered in the DB2 V9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.wn.doc/doc/i0052394.html>

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.migration.doc/doc/t0007187.html>

The arrangement of sections in the DB2 V9.5 Information Center have changed since the last version. Syntax and additional options for the commands we use here can now be found in **Database administration** → **Administrative interfaces** → **Commands**, under the subsections **CLP commands** and **System commands**. Otherwise, you can easily find information on each command with the **Search** function.

In order to minimize down time on your HADR servers, we recommend the version upgrade process illustrated in Figure 7-19.

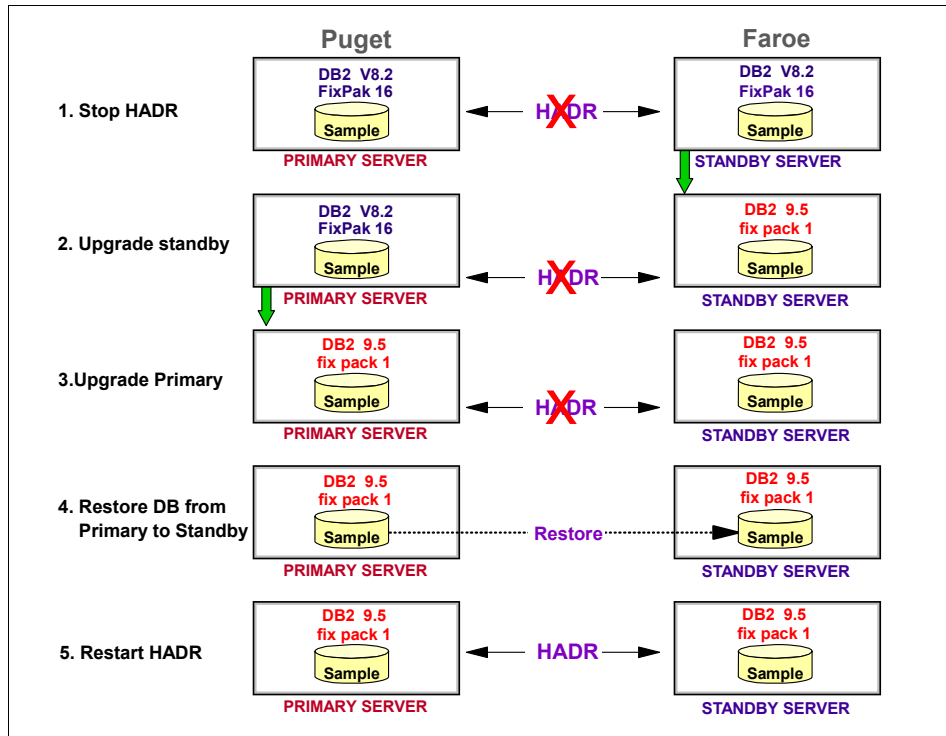


Figure 7-19 Version upgrade

## 7.2.1 DB2 Migration on Microsoft Windows GUI

Here we describe a step-by-step upgrade from DB2 Enterprise Server Edition version 8.2 fix pack 16 to version 9.5 fix pack 1, on a pair of Microsoft Windows servers. Our example HADR database pair name is SAMPLE.

All steps on Microsoft Windows are performed by the DB2 instance user (db2inst1 for our example) logged on with Local Administrator privileges. We illustrate all the dialog boxes for the DB2 V9.5 Installation on the HADR Standby server only, as they are the same as those we see on the HADR Primary server.

To help minimize downtime on the HADR Primary server, we execute the DB2 9.5 compressed package on both servers before stopping any DB2 instances. This extracts the installation files to a target directory (c:\temp\v95 in our example) in order to be ready to execute setup.exe, and db2ckmig.exe:

```
c:\temp\v95\SERVER\setup.exe
c:\temp\v95\SERVER\db2\Windows\utilities\db2ckmig.exe
(For DB2 ESE V9.1, the "SERVER" subdirectory is named "ESE")
```

We do this beforehand because the extraction process of almost 2000 files normally takes several minutes to perform. Also, this must be done so we can run the **db2ckmig** command on the V8.2 database to confirm that the migration to DB2 V9.5 will work without additional structural changes.

If setup.exe starts an Internet browser session that does not enable you to launch a DB2 installation, there might be execute permission restrictions in the browser settings. In this case, you can run a DB2 V9.5 installation directly from:  
c:\temp\v95\SERVER\db2\Windows\DB2 Server.msi

The equivalent DB2 V9.1 installation file would be named:

```
c:\temp\v91\ESE\db2\Windows\DB2 Enterprise Server Edition.msi
```

Our first set of upgrade steps involves no outage or downtime on the HADR Primary server. Any downtime activity here is limited to the HADR Standby server.

1. Upgrade the standby server.
  - a. Make sure that your HADR is in Connected, Peer state using **db2pd**, **get snapshot**, or the Manage HADR dialog. See Chapter 4, “HADR administration and monitoring” on page 99 for details on command usage.

Figure 7-20 shows our HADR Manage High Availability Disaster Recovery window for servers PUGET and FAROE.

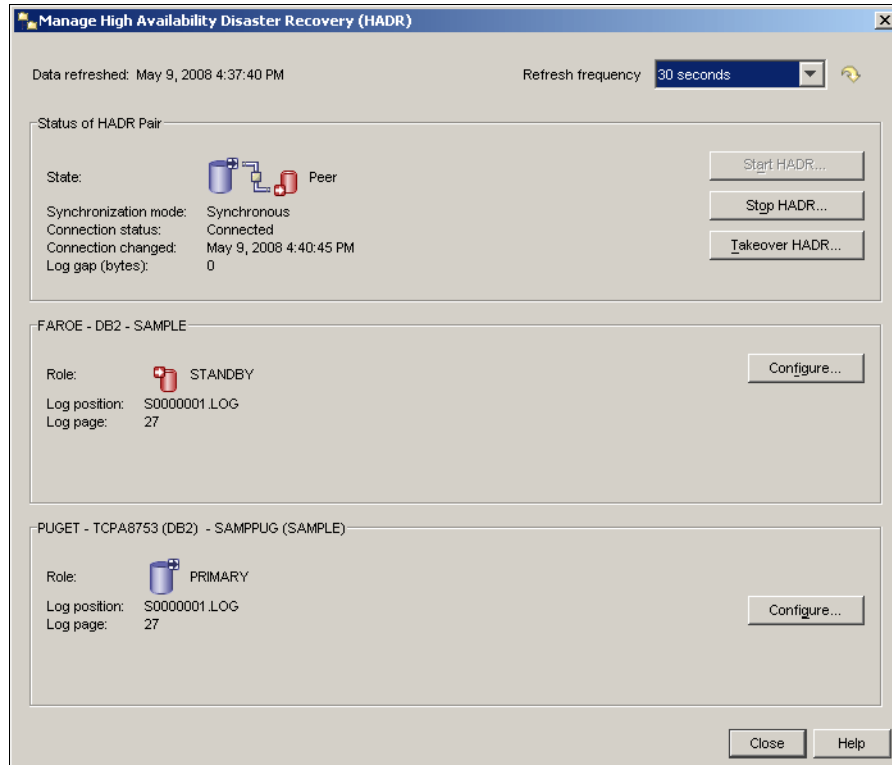


Figure 7-20 HADR Connected, Peer state before upgrade

- b. Run the **deactivate database** command on the CLP:
 

```
db2 deactivate db sample
```
  - c. You can reduce downtime further at this point by checking your database structure and content for eligibility prior to the upgrade downtime window using `db2ckmig.exe` from the `.\SERVER\db2\Windows\utilities` subdirectory of the extracted new DB2 version. This avoids added downtime on the primary database in later steps, at the cost of some added work before downtime.

Before running `db2ckmig`, we must:

- i. Stop HADR on the standby database:

```
db2 stop hadr on db sample
```

- ii. Disable remote user access to prevent split brain issues (change the server IP address or DB2 instance communication port number), for example:

```
db2 update dbm cfg using svccname 99999  
db2stop force  
db2start
```

- iii. Perform a **rollforward database** command:

```
db2 rollforward database sample complete
```

Now we can run **db2ckmig** from the extracted DB2 V9.5 utilities subdirectory:

```
db2ckmig sample -l c:\temp\db2ckmig.log
```

If the output is satisfactory, we can proceed to step d on page 199, otherwise corrective action must be taken on the HADR Primary database to make it eligible for upgrade. Output from a clean run of DB2 V9.5 **db2ckmig** looks like this:

```
db2ckmig was successful. Database(s) can be migrated.
```

After corrective action, either run **db2ckmig** on the HADR Primary, or proceed with the following steps for the HADR Standby if you want to avoid further downtime. We use the Command Line Processor (CLP) for brevity:

- i. Perform online database backup of HADR Primary with included logs:

```
backup db sample online to c:\temp\backup include logs  
compress
```

- ii. Copy the resulting database backup file to the standby server.

For Microsoft Windows servers, we simply share a common folder (*c:\temp* for example).

We copy our backup file across with Windows Explorer:

```
c:\temp\backup\SAMPLE.0\DB2\NODE0000\CATN0000\20080512\211945.  
001 :
```

- iii. Restore the database on the standby server, specifying a destination for the included log files:

```
db2 restore db sample from "c:\temp\backup" taken at  
20080512211945 logtarget "c:\temp\log" replace history file
```



iv. Roll forward to end of logs:

db2 rollforward database sample to end of logs and complete overflow log path (“c:\temp\log”)

v. Retry **db2ckmig** on the database, and reiterate this loop of steps after any further corrective action until the **db2ckmig** output log reports the database as eligible.

Step 1.j on page 202 refers to a warning to run **db2ckmig** in the normal sequence of upgrade events. We ignore it there because the **db2ckmig** command cannot be run on a database while it is in HADR Standby or rollforward pending mode. Also, we are going to replace it with a backup of the HADR Primary after we migrate that to a V9.5 database structure.

d. If you have not done so already in step c, stop the DB2 instance from the CLP or DB2 Control Center. From the Control Center, expand the object tree down to the database object, right-click the database name, then click **Stop**. This will stop the instance as well as deactivating the database.

e. Close the DB2 Control Center and any other DB2 GUI tools or windows, including the db2systray.exe icon in the System Tray if it is present.

f. Stop all DB2 prefixed services in the Microsoft Management Console Services Snap-in:

Click **Start** → **Run**, type **services.msc**, then press Enter.

Right-click each DB2-prefixed service that has a Status of Started and click **Stop**.

g. Open **setup.exe** from the DB2 V9.5 installation source directory:

c:\temp\v95\SERVER\setup.exe

h. The first dialog box to appear is Figure 7-21 - Install a Product.

DB2 V9.5 has option buttons to Install New or Work with Existing. Click **Work with Existing**.



Figure 7-21 DB2 V9.5 - Install a Product

i. Work with an Existing DB2 Copy.

Select the DB2 copy to work with, shown in Figure 7-22.

When migrating from a DB2 V8, there will not be a DB2 Location defined yet. DB2 V9.5 assumes a location name of DB2V8 for the old existing installation code, using the current DB2 installation path and an Action of Migrate. Accept the defaults and click **Launch DB2 Setup wizard**.

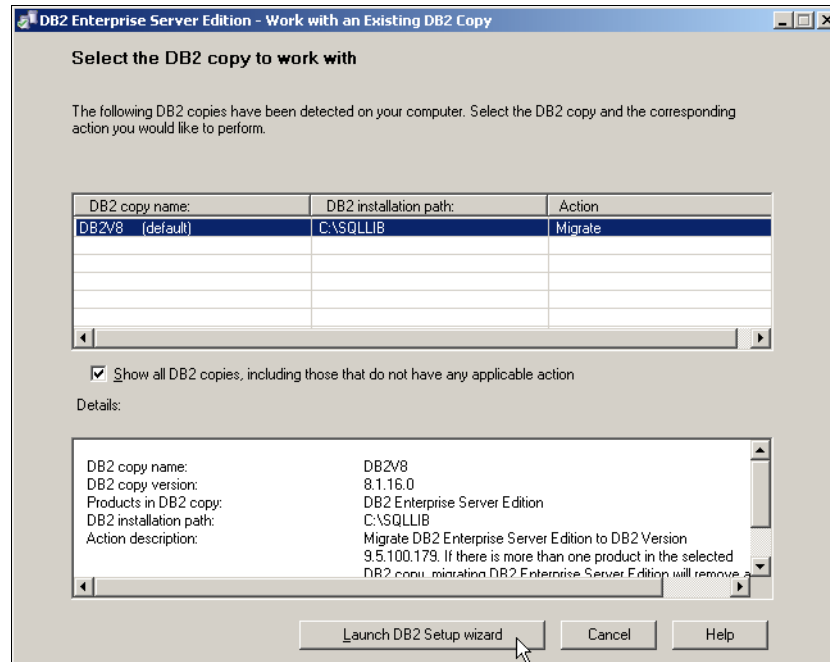


Figure 7-22 Work with an Existing DB2 Copy

- j. Warning dialog box for existing databases, and removal of all pre-DB2 V9.5 products, shown in Figure 7-23. Click **OK** to accept this and proceed.

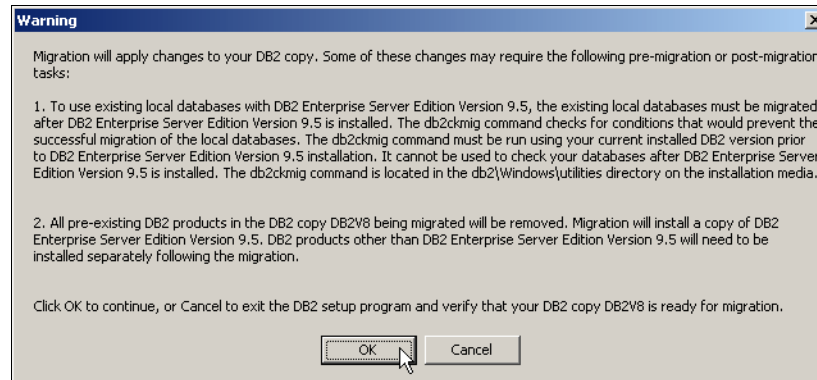


Figure 7-23 Warning - removal of all pre-DB2 V9.5 products

- k. Windows Installer - Preparing to install..., shown in Figure 7-24. Wait for this to complete (it might take a minute) before the next stage.

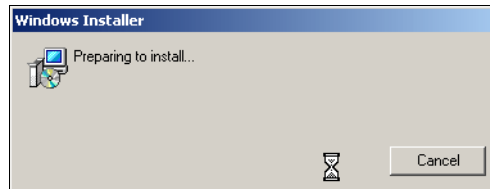


Figure 7-24 Preparing to install

I. DB2 Setup - DB2 Enterprise Server Edition - DB2COPY1

Figure 7-25 shows the welcome panel for the DB2 Setup wizard. Click **Next**.

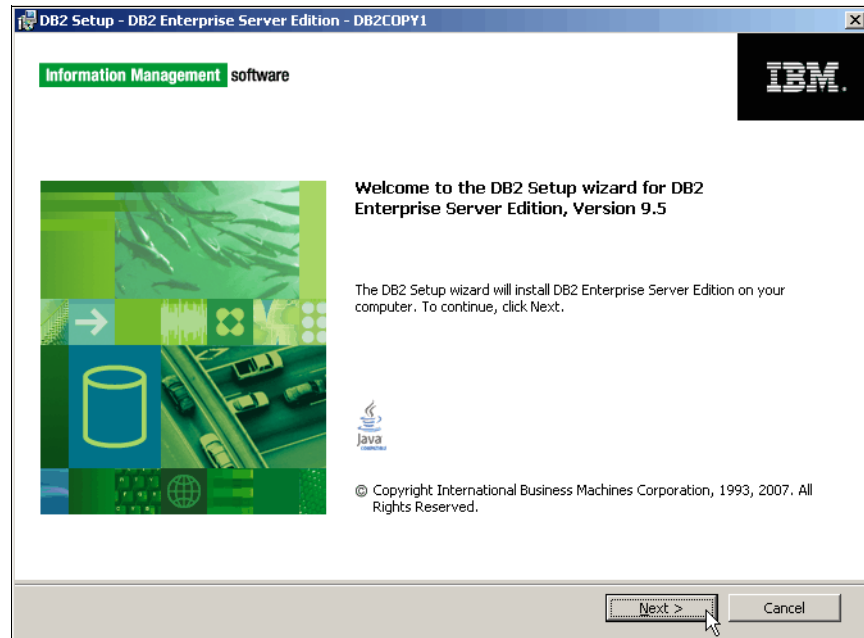


Figure 7-25 DB2 ESE V9.5 Setup wizard

m. Software Licence Agreement, shown in Figure 7-26.

Click the radio button I accept the terms in the licence agreement and click **Next**.

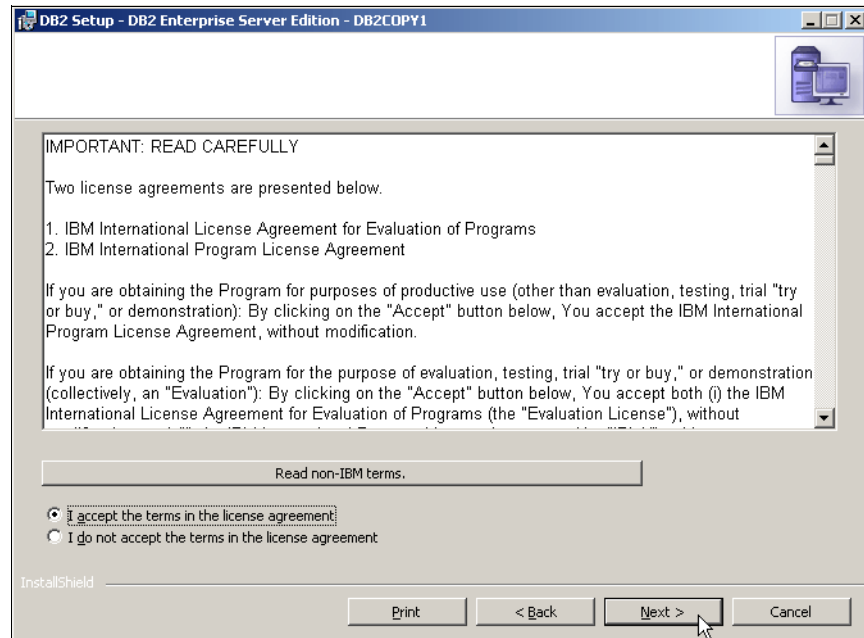


Figure 7-26 Software licence agreement - accept it

n. Select the installation type, shown in Figure 7-27.

Click a radio button to choose your install type (Typical, Compact, or Custom). (We choose **Typical** for our test lab). Click **Next**.

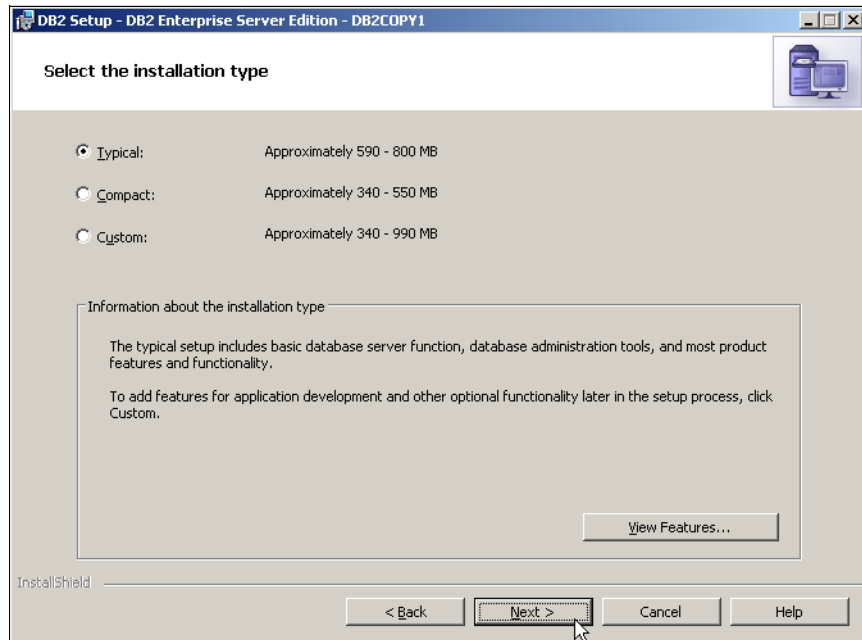


Figure 7-27 Select the installation type

- o. Select the Installation, response file creation, or both, in Figure 7-28. Click a radio button to choose Install..., Save..., or do both. We choose to do both. Click **Next**.

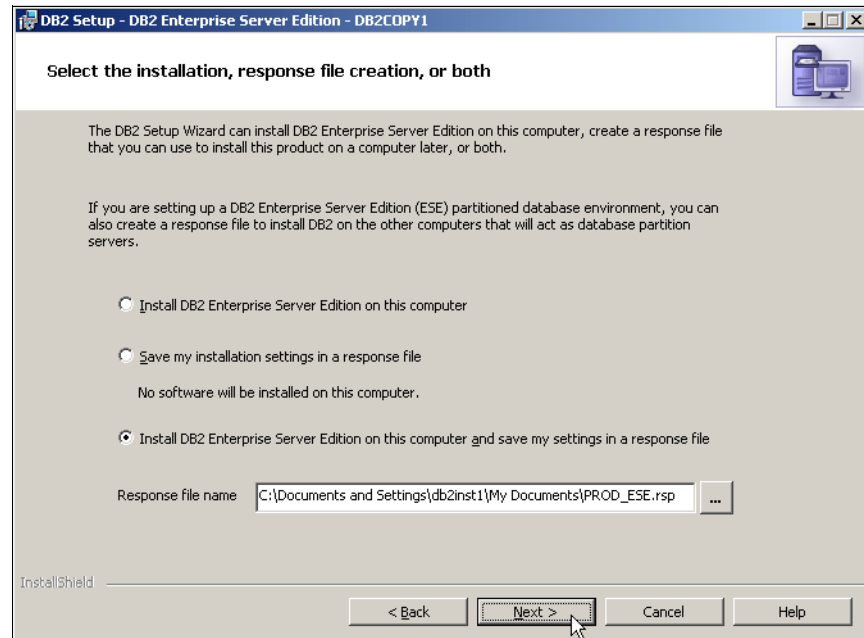


Figure 7-28 Select installation, response file creation, or both



p. The Installation folder is shown in Figure 7-29.

For a Typical install type, this option is disabled and defaults to the existing DB2 installation folder. Click **Next**.

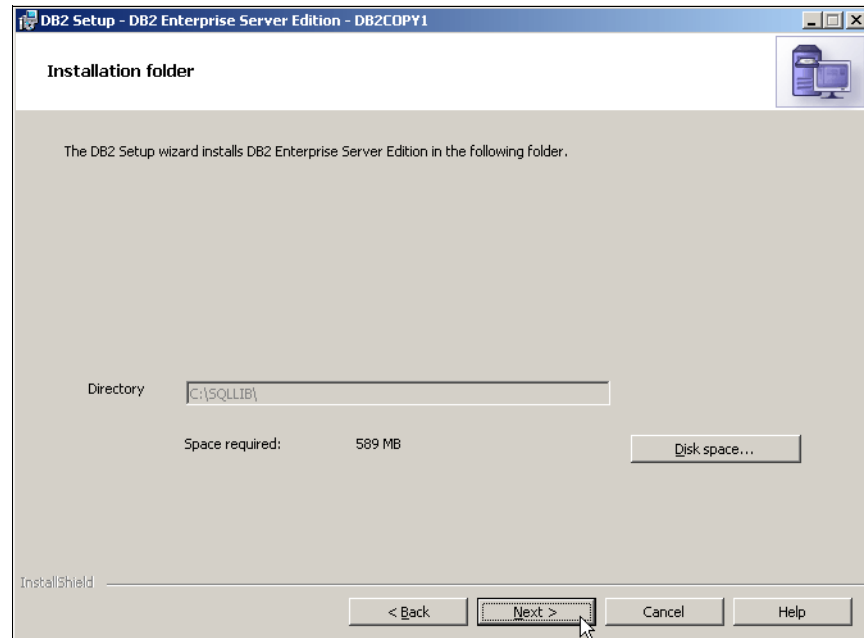


Figure 7-29 Installation folder (disabled for Typical install type)

q. Set the DB2 copy name, shown in Figure 7-30.

Choose a level of abstraction for the destination folder for a given DB2 version or fix pack (for co-existence of multiple DB2 versions). We accept the default DB2COPY1. Click **Next**.

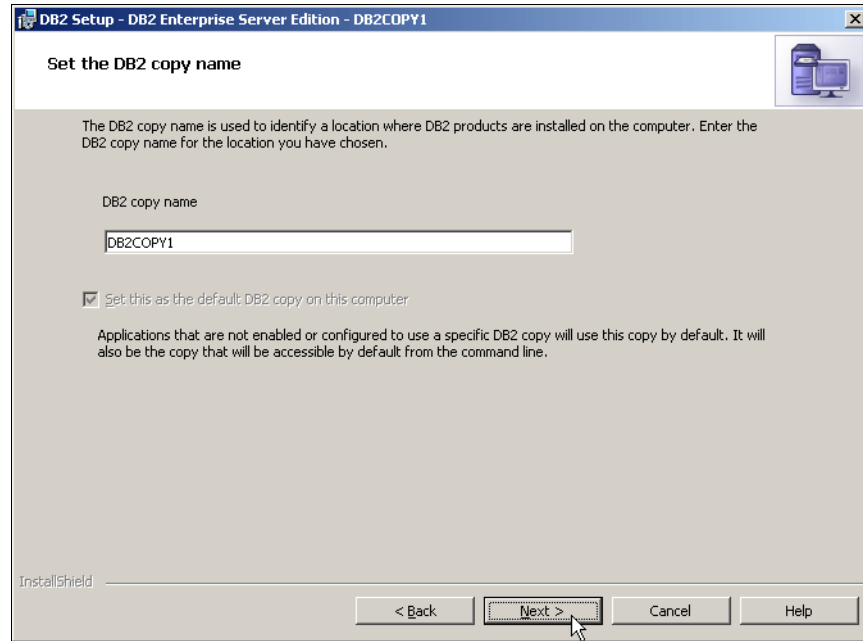


Figure 7-30 Set the DB2 copy name

- r. Set user information for the default DB2 instance, in Figure 7-31. Specify the Domain, User name, Password, Confirm password. We choose to use db2inst1 as the User name rather than the default db2admin. Click **Next**.

DB2 Setup - DB2 Enterprise Server Edition - DB2COPY1

Set user information for the default DB2 instance

Specify the required user information that the DB2 instance, DB2, will use to log on to your system.

User information

Domain: None - use local user account

User name: db2inst1

Password: \*\*\*\*\*

Confirm password: \*\*\*\*\*

InstallShield

< Back Next > Cancel Help

Figure 7-31 Set user information for the default DB2 instance

**Note:** If you do not want to create a Domain-based user ID or you specifically want DB2 services controlled by local user IDs, leave the Domain field blank.

- s. Enable operating system security for DB2 objects, in Figure 7-32.
- Optional — this alters the Microsoft Windows filesystem Access Control List (ACL) so that only users in the DB2 groups can access the DB2 files/folders). We deselect the check box, choosing not to enable. Click **Next**.

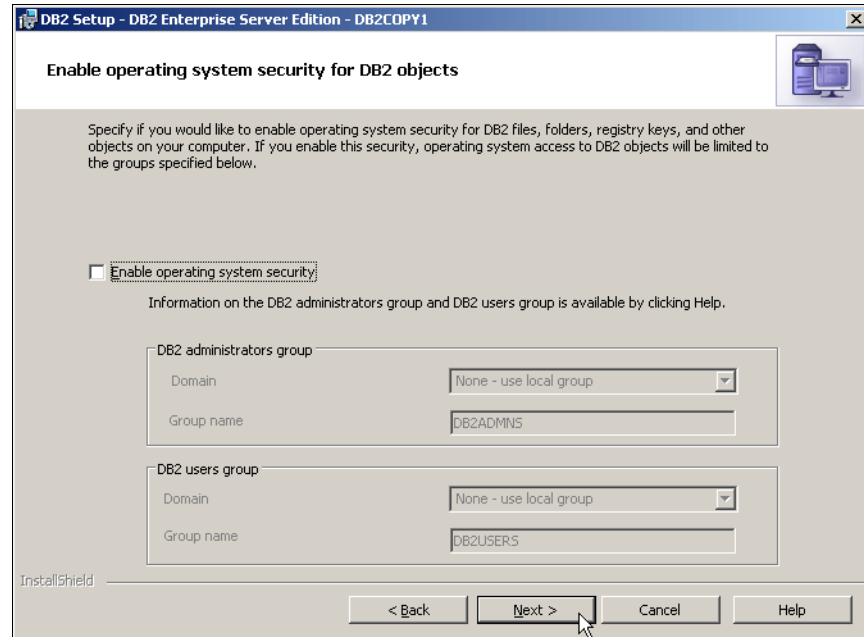


Figure 7-32 Operating system security for DB2 objects

- t. Start copying files and create a response file, in Figure 7-33.  
Review your Current Settings and click **Finish**.

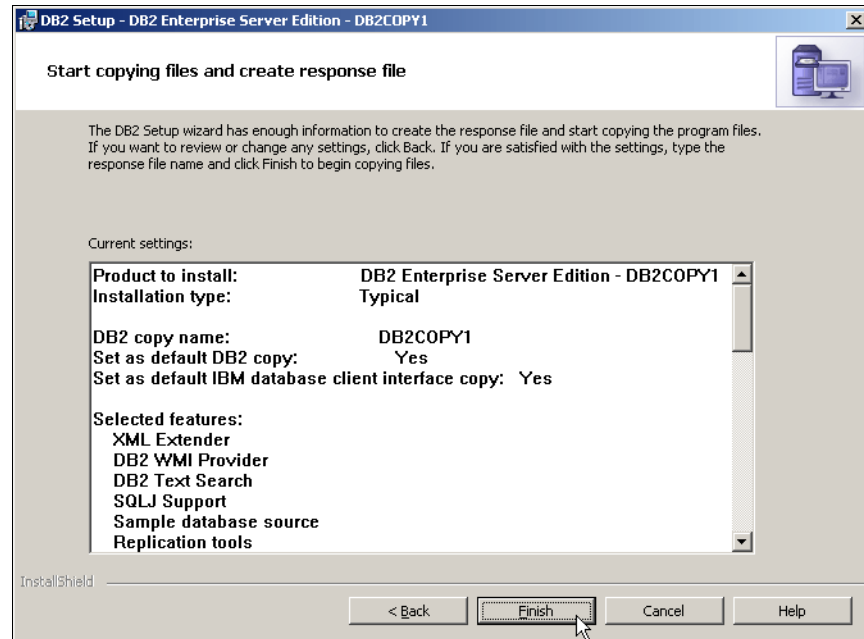


Figure 7-33 Start copying files and create response file

- u. Installing DB2 Enterprise Server Edition - DB2COPY1, in Figure 7-34.  
Watch the progress bar as files are copied and installation steps are performed in the background. This will take several minutes to complete. Wait for the next dialog box to appear.

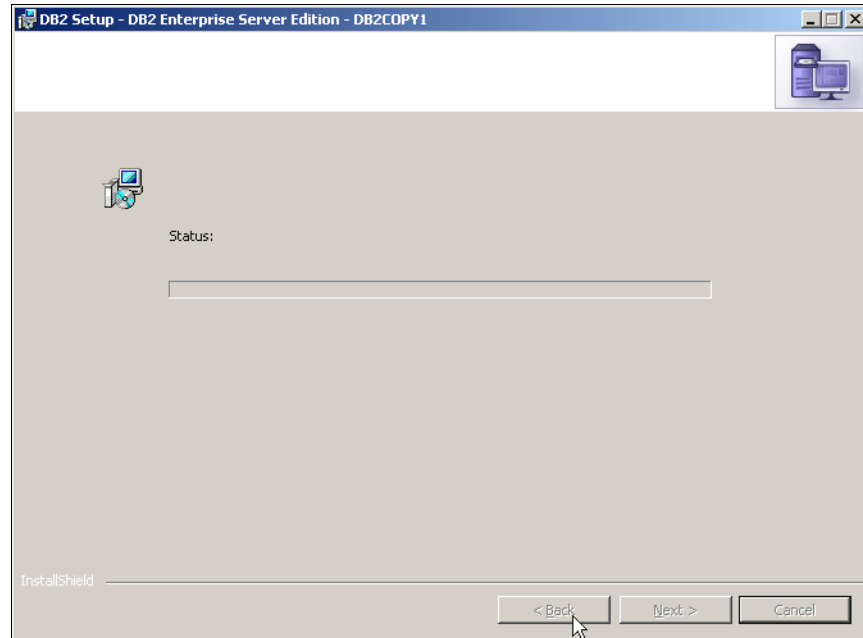


Figure 7-34 Setup progress bar

v. Setup is complete, in Figure 7-35.

There is a recommendation to complete post install steps. Review for your information and click **Next**.

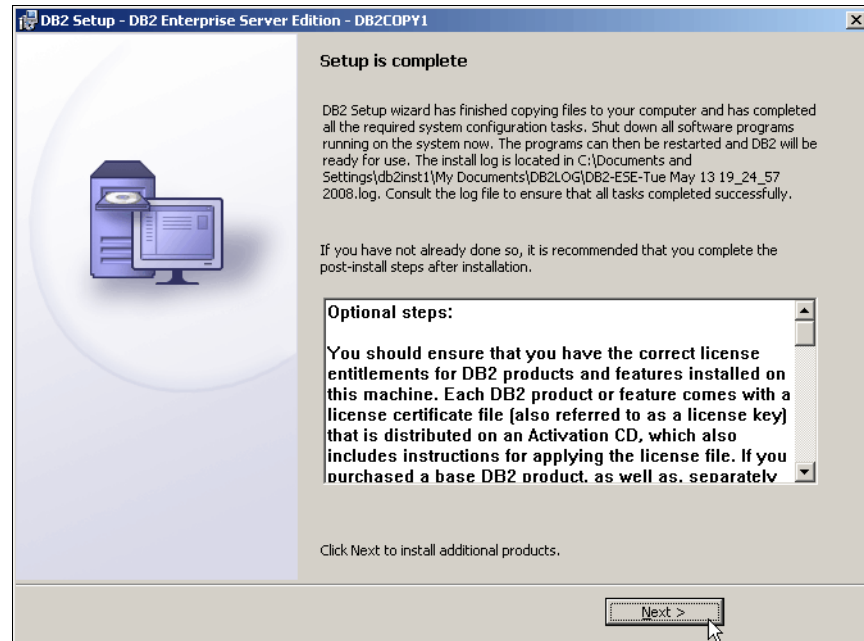


Figure 7-35 Setup is complete - well, almost...

w. Install additional products, in Figure 7-36.

Optionally install IBM Database Add-Ins for Visual Studio® - We choose not to do this, and just click **Finish**.

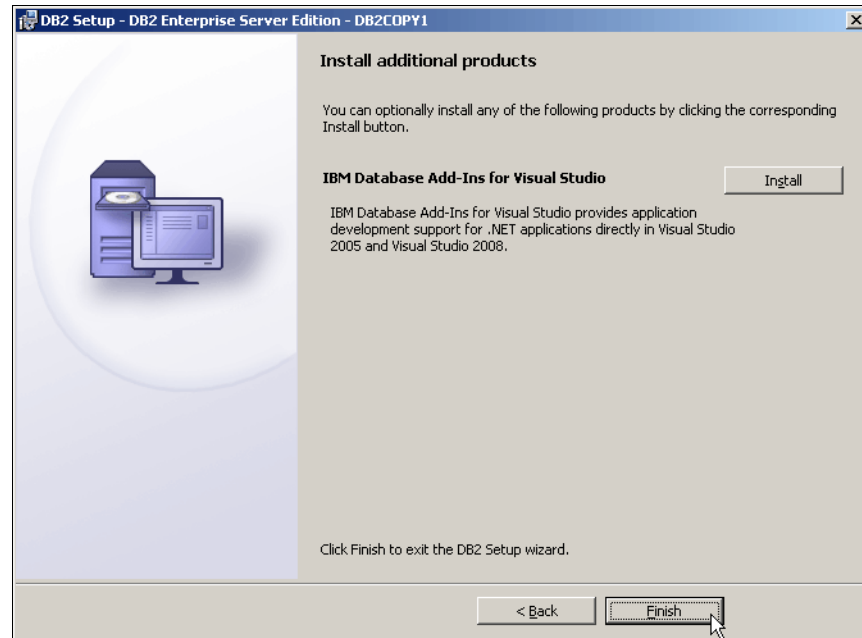


Figure 7-36 Install additional products



x. The First Steps window appears, shown in Figure 7-37.

Review for your information and click **Exit**.

DB2 First steps can be run again at any time from the Microsoft Windows Menu:

**Start → All Programs → IBM DB2 → DB2COPY1 (Default) → Set-up Tools → First Steps**

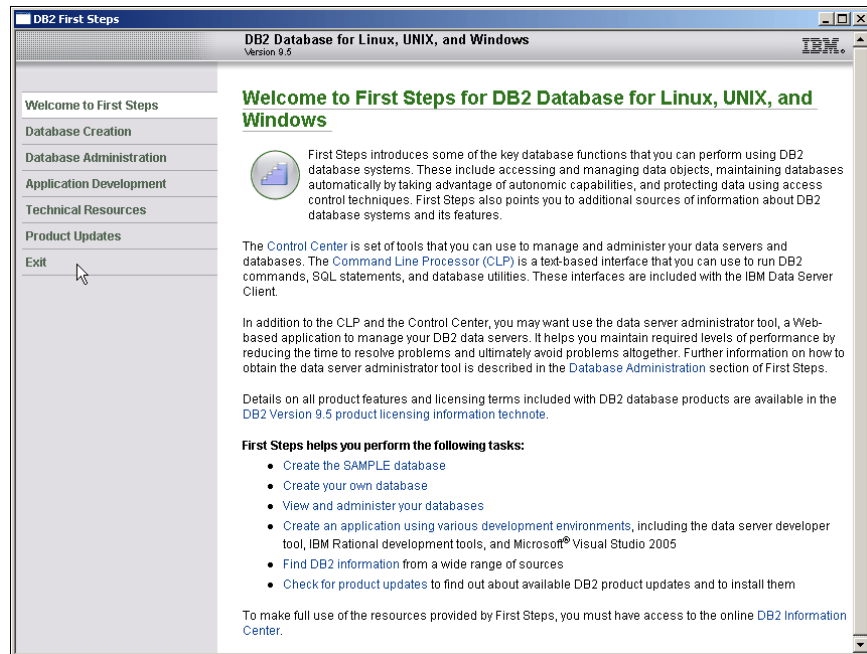


Figure 7-37 Welcome to First Steps for DB2

y. We confirm our new DB2 version with the **db2level** command:

```
C:\SQLLIB\BIN>db2level
DB21085I Instance "DB2" uses "32" bits and DB2 code release
"SQL09051" with
level identifier "03020107".
Informational tokens are "DB2 v9.5.100.179", "s080328",
"WR21402", and Fix Pack
"1".
Product is installed at "C:\SQLLIB" with DB2 Copy Name
"DB2COPY1".
```

The DB2 V9.5 CLP is reached through a similar menu path as for the old V8.2:

**Start** → **All Programs** → **IBM DB2** → **DB2COPY1(Default)** → **Command Line Tools** → **Command Window**

The only difference is the addition of the new DB2COPY1 Location, to allow co-existence of multiple versions of DB2 in the one server. In our example, **DB2COPY1(Default)** is what we accepted as the DB2 Location in the installation process.

2. Upgrade the primary server:

Log on to the primary server (PUGET in our test case) with the DB2 instance user, which should be a Local Administrator. Because of the different version levels of your two servers, you will not be able to switch roles to the standby, so your database down-time for users starts here.

a. Arrange for all DB2 connections (users, applications, batch) to stop. This will avoid the need to issue a **force applications** command, causing potentially lengthy rollback actions. Removal of connections can be measured with a **db2 list applications** command issued from the CLP.

b. From the CLP, issue the **deactivate database** command:

```
db2 deactivate database sample
```

The deactivate database command brings your database into a consistent state in preparation for the migration check and later structural conversion.

c. If you ran the **db2ckmig** command in step 1.c on page 197 and the structure of the HADR Primary database has not changed since then, it is not necessary to run **db2ckmig** here, you can skip to step d.

Otherwise, run **db2ckmig** on the HADR primary database, and confirm the output is satisfactory and the database is eligible for upgrade:

```
db2ckmig sample -l c:\temp\db2ckmig.log
```

d. Perform a backup of your database (we recommend offline for ease of restoration without logs or rolling forward). This can be done either through the Control Center Backup wizard or CLP. To issue the backup from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Backup**, as shown in Figure 7-38. This step is performed in case migration fallback is needed. Even if the database migration is successful, there might be application dependencies that require lengthy corrective action and testing before re-migration to the new DB2 version is practical.

```
db2 backup database sample to c:\temp\backup compress
```

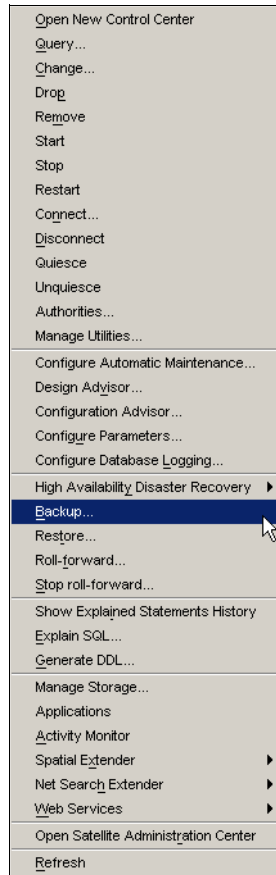


Figure 7-38 Right-click the database in Control Center, then left-click Backup

- e. Run the **db2support** command with collection level zero, to get instance config information and other information for fallback if necessary.

```
db2support c:\temp -cl 0
```

Run the **db2cfexp** command to save node and database directory information in case the DB2 product code needs to be un-installed and re-installed.

```
db2cfexp c:\temp\nodedb.cfg backup
```

After any fallback requiring reinstallation of DB2 product code, the **db2cfimp** command can be used to re-import these definitions where necessary with syntax such as:

```
db2cfimp c:\temp\nodedb.cfg
```

- f. Perform steps 1.d on page 199, through to 1.y on page 215 to install the new DB2 version on the HADR Primary server.

The default DB2 instance is automatically migrated on Microsoft Windows as part of these steps. Any extra instances can be migrated later with the **db2imigr** command.

The **db2imigr** command implicitly calls **db2ckmig** on all databases in that instance, and will not migrate the instance if any errors are found. This emphasizes the need to explicitly perform **db2ckmig** well in advance to ensure no failures at a more critical stage.

The **db2imigr** command is described in:

<http://localhost:51002/help/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002055.html>

- g. Issue the **migrate database** command on all local databases in the new V9.5 DB2 instance(s). Note that this also applies to any system tools database (used by the DB2 Task Scheduler). We migrate our HADR database with:

```
db2 migrate database sample
```

Successful completion results in the following message:

```
DB20000I The MIGRATE DATABASE command completed successfully.
```

The **migrate database** command is covered in:

<http://localhost:51002/help/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0001959.html>

At this stage, you can consider the actual DB2 migration process completed, and application connectivity testing can commence. The next steps can be performed in parallel with user testing before considering your system ready for production.

3. Re-establish HADR database pairing between Standby server (FAROE) and Primary server (PUGET).
  - a. Perform an online backup of your HADR Primary database, which is copied over to the Standby server in order to re-establish HADR. As in step 2.d, you have the option to use the Control Center Backup wizard, or the CLP as we do here:

```
db2 backup database sample online to c:\temp\backup compress
```
  - b. Copy the backup image over to the HADR Standby server ready to restore. Here we note that finally, DB2 V9 on Windows is using the same file naming convention as for UNIX platforms, in that the backup files are not in a long tree of subdirectories, but are long filenames in the designated subdirectory. We use our shared folder (*c:\temp*), and copy our backup file across with Windows Explorer:

```
c:\temp\backup\SAMPLE.0.DB2.NODE0000.CATN0000.yyyymmddhmmss.001
```

- c. Log on to the Standby server.
- d. Restore the database from the backup image copied across in step 3.b. You can either use the Control Center Restore wizard or the **restore** command on the CLP. To access the Restore wizard from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Restore**. Our CLP syntax is:

```
db2 restore database sample from C:\temp\backup taken at
yyymmddhhmss
```

We accept the request to overwrite the existing database. Successful output from this command when run against an existing V8.2 database includes an extra message:

```
SQL2555I The restored database was successfully migrated to the
current release.
```

This indicates that our pre-existing standby database has been migrated as part of the restore.

- e. Update database configuration parameters for HADR Standby. Normally, if we had simply restored the primary database over the existing unchanged old HADR standby database, no action would be necessary here, because existing database configuration parameters for HADR Standby are retained.

We experienced that after our V9.5 migration with restore, the database configuration parameters had reverted back to the HADR Primary settings.

This is why it is imperative to check that the HADR Standby database configuration parameters are correct, and update any as necessary.

We use this CLP command syntax to get the current settings:

```
db2 get db cfg for sample |findstr "HADR"
```

We expect four key HADR Primary database and Standby database configuration parameters of our example to be as shown in Table 7-1.

*Table 7-1 Expected Primary and Standby HADR db cfg values*

	Primary	Standby
HADR local host name (HADR_LOCAL_HOST)	PUGET	FAROE
HADR local service name (HADR_LOCAL_SVC)	55001	55002
HADR remote service name (HADR_REMOTE_HOST)	FAROE	PUGET
HADR remote service name (HADR_REMOTE_SVC)	55002	55001

Our Standby db cfg values need to be reset using the following commands:

```
db2 update db cfg for sample using hadr_local_host FAROE
db2 update db cfg for sample using hadr_local_svc 55002
db2 update db cfg for sample using hadr_remote_host PUGET
db2 update db cfg for sample using hadr_remote_svc 55001
```

- f. If you disabled remote user access in step 1.c on page 197 by changing the HADR Standby server IP address or DB2 instance communication port number, then reset it back to the original value. We set the communication port back to our original setting of 50000:

```
db2 update dbm cfg using svcname 50000
db2stop
db2start
```

- g. Start HADR on standby and primary databases.

Our standby database was taken out of the HADR role of STANDBY and rolled forward in step 1.c on page 197 in order to run db2ckmig. The current HADR role value is still STANDARD, and we change it back to STANDBY by starting HADR on that database. The primary database will also be in STANDARD role, and we change it to PRIMARY role with another **start hadr** command.

Commands to start HADR can be issued remotely using the CLP or Control Center GUI, so long as a remote database alias has been cataloged for the other database in the HADR pair. It is recommended to start HADR on the standby first, then the primary, which is what the Control Center GUI will always do, and what you should also do using the CLP unless you can guarantee that you would be able to start the standby within the HADR\_TIMEOUT value after starting the primary.

We present the following examples for your reference, so you can choose the technique you prefer:

i. Using CLP on the standby server:

```
db2 deactivate db sample
db2 start hadr on db sample as standby
db2 deactivate db samppug user db2inst1 using passblah
db2 start hadr on db samppug user db2inst1 using passblah as
primary
```

Starting the HADR Primary database remotely from the standby server is possible when there is a remote database alias cataloged. Our standby server has a remote database alias for SAMPLE cataloged as SAMPPUG, pointing at a cataloged node definition for our PUGET server. Otherwise, these CLP commands can be split up so that the first two commands are run against the local SAMPLE database on the standby server, and then the latter two are run locally from the primary server, without any need to specify user and password.

ii. Using CLP on the primary server:

```
db2 deactivate db sampfar user db2inst1 using passblah
db2 start hadr on db sampfar user db2inst1 using passblah as
standby
db2 deactivate db sample
db2 start hadr on db sample as primary
```

As noted in the previous option, Starting the HADR Standby remotely from the primary server is possible when there is a remote database alias cataloged. Our primary server has a remote database alias for SAMPLE cataloged as SAMPFAR, pointing at a cataloged node definition for our FAROE server.

iii. Using the Control Center:

To start HADR from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Manage...** as shown in Figure 7-39.

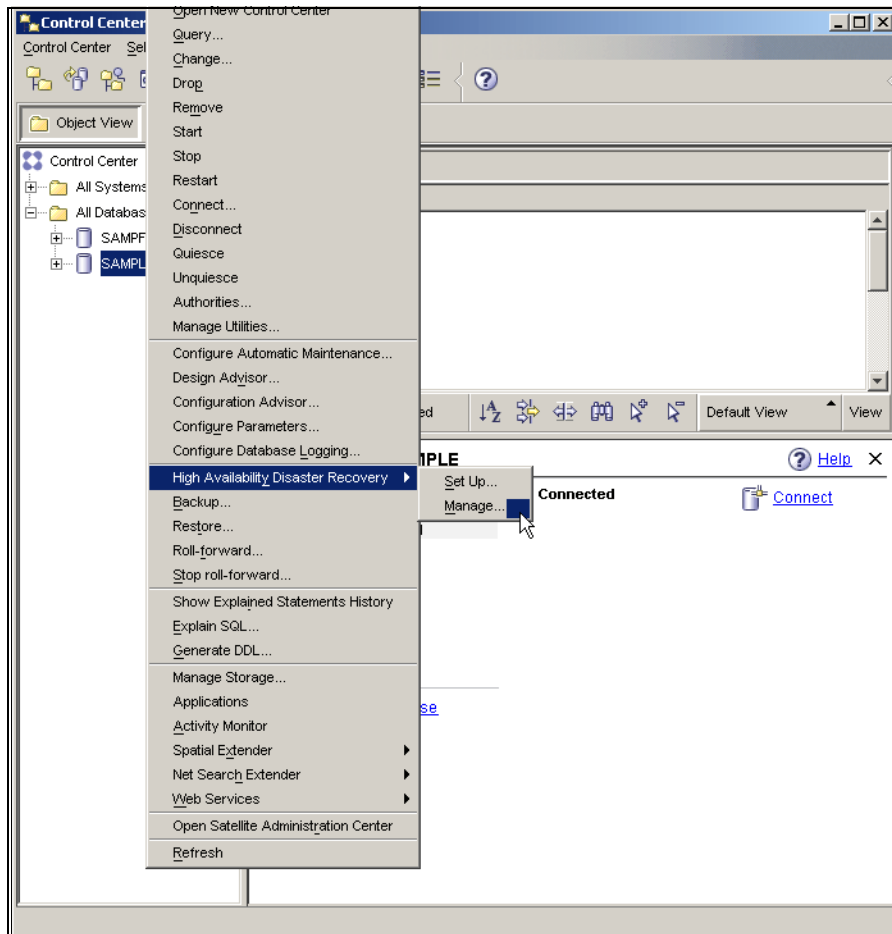


Figure 7-39 Managing HADR from Control Center

In the next dialog box shown in Figure 7-40, we can see that both databases are in STANDARD role. If you start the Manage HADR dialog on the standby server, the remote database might not be cataloged. In this case, you can click **Catalog Database** to start a wizard to catalog and assign the remote database alias for use in the Manage HADR dialog, which might require the Control Center to be stopped and restarted to detect the new database entry.



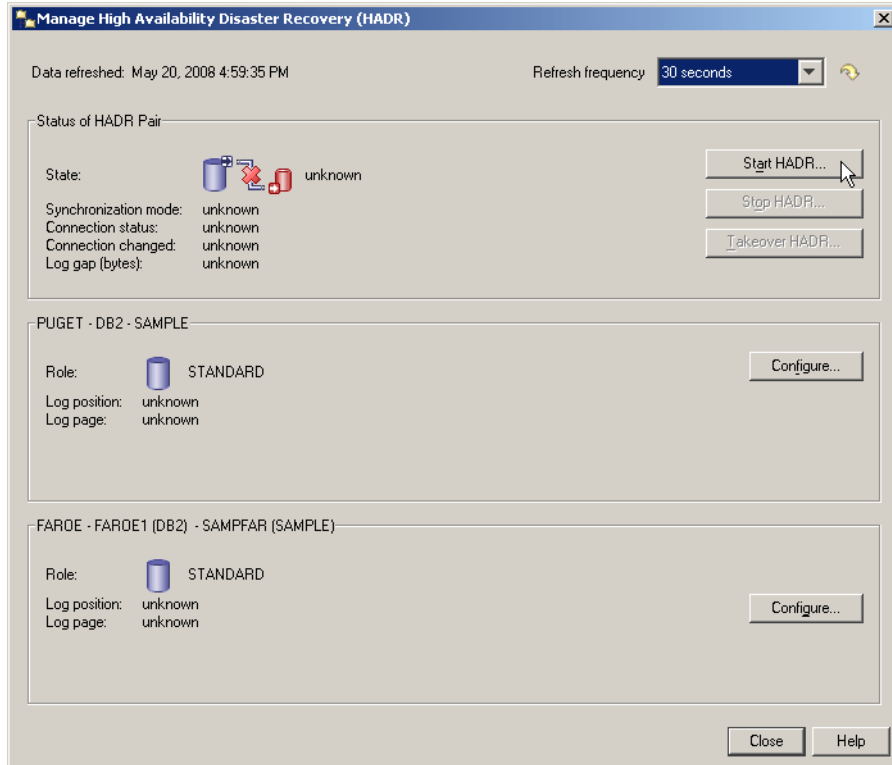


Figure 7-40 Manage HADR dialog - about to Start HADR

Click **Start HADR**, and accept the default settings as shown in Figure 7-41. Click **OK** to run the command, optionally clicking **Show Command** first to compare it to the CLP process we show in option i and ii.

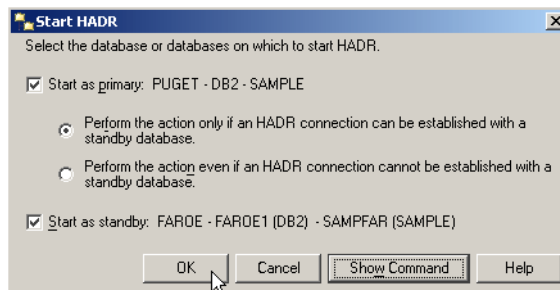


Figure 7-41 Default options for start HADR dialog box

HADR should now be started on the primary and standby servers, and in Connected, Peer state within a very short time-frame.

All users and applications can now connect to the migrated database on the HADR Primary server in full production mode, safe in the knowledge DB2 has a working hot backup with the HADR Standby database.

There is a reason for the recommendation to start HADR on Standby first, and Primary second. A Standard role database will not normally activate in HADR Primary role without an HADR Standby being activated and successfully communicating within the HADR\_TIMEOUT period. An HADR Standby database, on the other hand, can be started without an HADR Primary being present or active. An eligible Standard role database can only be started as HADR Primary without a connection to an active HADR Standby using the **by force** parameter of the **start hadr** command.

If a database has active connections, and a **start hadr** command is issued against it, this can fail with error code SQL1767N or SQL1768N, depending on the initial state of the database. This is why the **deactivate database** command precedes all **start hadr** commands issued by the Control Center GUI. Similarly, error code SQL1769N is issued if the **deactivate database** command does not precede a **stop hadr** command on an active HADR Standby database. The DB2 V9.5 Information Center contains a useful table on how a database in any given state behaves when a **start hadr** command is issued against it:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0011552.html>

Here is the equivalent link for **stop hadr**:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0011551.html>

This concludes our description of a DB2 migration using GUI tools in an HADR configuration to minimize downtime in a production environment.

## 7.2.2 DB2 migration on Linux command line

This section covers an example of a DB2 migration of an HADR database server pair using the Command Line Processor (CLP), to help provide minimal downtime in a production environment. In our test case we migrate DB2 Data Server (Enterprise Server Edition) from version 9.1 to version 9.5 on a Linux platform. The servers used in our example are LEPUS and MENSA, both using the database SAMPLE, with instance user “db2inst1”, and DB2 Admin Server user “db2as”.

Some initial steps are performed by the root user and DB2 Admin Server user and are specifically highlighted. All other steps are performed by the DB2 instance user (db2inst1 for our example). While there are choices to perform

most steps with a DB2 GUI, this section only shows the command line method, as many sites perform remote administration of UNIX servers through text-only session emulators rather than bandwidth-hungry GUI emulation of X11 windows.

A description of the process of migration to DB2 V9.5 can be found in the following Information Center link:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.migration.doc/doc/t0007200.html>

## Procedure

To help minimize downtime on the HADR Primary server, we extract the DB2 V9.5 fix pack 1 compressed package on both servers, before stopping any DB2 instances:

```
tar -xvpf v9.5fp1_linuxx64_server.tar.gz
```

This extracts the installation files to a target directory (/db2temp in our example) in order to be ready to execute **db2\_install**

```
/db2temp/server/db2_install
```

Unlike on the Microsoft Windows platform, there is no **db2ckmig** file in a subdirectory of extracted installation source for UNIX platforms. The DB2 product installation to target directories must be performed first. The recommended process for migration is execute the **db2\_install** for DB2 V9.5 into a new target location without impacting the existing DB2 product code, and then gradually migrate all old instances across to the new location. This means that we execute **db2\_install** on both servers right now without requiring any downtime.

After **db2\_install** has been performed using a default installation target, the DB2 V9.5 **db2ckmig** command is found here:

```
/opt/ibm/db2/V9.5/bin/db2ckmig
```

We do these extract and install steps beforehand, as the extraction process and product code installation normally takes several minutes to perform, and it confirms the integrity of the downloaded package. We also need this done so we can run the V9.5 **db2ckmig** command on the standby server's V9.1 database, before any downtime window. This confirms that the migration will work without additional structural changes.

The following steps outline DB2 migration, first on the HADR Standby server, then on the Primary, and finally, re-establishing HADR pairing.

1. Start the procedure on the standby server (MENSA). This first subset of upgrade steps involves no outage or downtime on the HADR Primary server - any downtime activity here is limited to the HADR Standby server.

- a. Confirm HADR is in Connected, Peer state:

```
db2pd -d sample -hadr
```

We expect to see something similar to Example 7-2:

*Example 7-2 Output of db2pd showing HADR Connected, Peer state*

---

```
db2inst1@lepus:~/script> db2pd -d sample -hadr
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days  
00:00:21
```

HADR Information:

Role	State	SyncMode	HeartBeatsMissed
Primary	<b>Peer</b>	Sync	0

LogGapRunAvg (bytes)

0

ConnectStatus	ConnectTime	Timeout
<b>Connected</b>	Thu May 22 01:28:43 2008 (1211444923)	120

LocalHost	LocalService
lepus	55101

RemoteHost	RemoteService
RemoteInstance mensa db2inst1	55102

PrimaryFile	PrimaryPg	PrimaryLSN
S0000002.LOG	0	0x0000000001770000

StandByFile	StandByPg	StandByLSN
S0000000.LOG	0	0x0000000000FA0000

---

- b. Deactivate the database and stop HADR:

```
db2 deactivate database sample  
db2 stop hadr on database sample
```

- c. Unless you can guarantee that no remote DB2 connections to the standby server can occur, disable remote user access to prevent split brain issues. We recommend changing the server IP address or DB2 instance communication port number, for example:

```
db2 update dbm cfg using svcname 99999  
db2stop force  
db2start
```

d. In this step you have a choice to roll forward the HADR Standby database to bring it into a consistent state to perform **db2ckmig**. If we do not roll forward this database, we need to drop it before performing the db2imigr command in the later step g on page 227.

i. Performing a **rollforward database** on the standby and running **db2ckmig** on it enables us to avoid unnecessary and unforeseen downtime on the primary, in the event of errors in the output of **db2ckmig**. Iterative loops of corrective action on the database and **db2ckmig** commands can then be performed without requiring any downtime on the primary, recording any useful corrective actions for later. After a successful migration on the standby is confirmed, those recorded corrective actions can be mirrored on the primary inside the downtime window. We use the following rollforward syntax:

```
db2 rollforward database sample complete
```

ii. Because we have delivered the DB2 V9.5 product code to the target libraries with the db2\_install command, we can run db2ckmig using the current DB2 instance user against our standby database:

```
/opt/ibm/db2/V9.5/bin/db2ckmig sample -1 db2ckmig_sample.txt
```

Descriptions of the syntax and options of the db2ckmig command can be found in the DB2 V9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.1uw.admin.cmd.doc/doc/r0002028.html>

e. Stop the DB2 instance:

```
db2stop force
```

f. Stop the DB2 Admin Server from the DAS user id:

```
db2admin stop
```

g. Migrate the DB2 instance:

This step will implicitly call db2ckmig on all databases within the instance.

As the root user, we issue the db2imigr command:

```
/opt/ibm/db2/V9.5/instance/db2imigr db2inst1
```

We expect to see the following output from this:

```
db2ckmig was successful. Database(s) can be migrated.
```

```
DBI1070I Program db2imigr completed successfully.
```

At this stage, our work on the standby server is done - DB2 is now at V9.5, which we can confirm with a **db2level** command as shown in Example 7-3.

*Example 7-3 db2level output shows our new DB2 version.*

---

```
db2inst1@lepus:~> db2level
DB21085I Instance "db2inst1" uses "64" bits and DB2 code release
"SQL09051"
with level identifier "03020107".
Informational tokens are "DB2 v9.5.0.1", "s080328", "MI00225", and Fix Pack
"1".
Product is installed at "/opt/ibm/db2/V9.5".
```

---

2. These are the steps to execute on the primary server. At this point you will have an outage on the primary database:
  - a. Arrange for all DB2 connections (users, applications, batch) to stop. This will avoid the need to issue a **force applications** command, causing potentially lengthy rollback actions.
  - a. Deactivate and stop HADR on the primary database:

```
db2 deactivate database sample
db2 stop hadr on database sample
```
  - b. Stop the DB2 Instance:

```
db2stop
```
  - c. Stop the DB2 Admin Server from the DAS user id:

```
db2admin stop
```
  - d. Migrate the DB2 instance:

Because we have already placed the DB2 V9.5 code into the target location, we have the root user issue the **db2imigr** command from that location:

```
/opt/ibm/db2/V9.5/instance/db2imigr db2inst1
```

After the db2imigr command, the symbolic links to DB2 product code for the DB2 instance user will have been updated to point to the new DB2 version. These can be found in our /home/db2inst1/sqllib subdirectory. All subsequent commands issued by our DB2 instance user are V9.5 commands.
  - e. Start the DB2 instance:

```
db2start
```
  - f. Start DB2 Admin Server from the DAS user:

```
db2admin start
```
  - g. Migrate the database:

```
db2 migrate db sample
```

At this stage, DB2 on the primary server is ready for initial application testing.

3. The following steps restore the standby database and re-establish the HADR pairing:

a. On the primary server, back up the database.

```
db2 backup database sample online to /db2/backup compress
```

b. Move the backup image over to the standby server. We use the example Secure Shell Copy (scp):

```
scp SAMPLE.0.db2inst1.NODE0000.CATN0000.yyyymmddhhmmss.001
db2inst1@mensa:/db2/backup
```

c. Log on to the standby server.

d. Restore the database on the standby server:

```
db2 restore database dbname from /db2/backup taken at
yyyymmddhhmmss
```

e. If you changed the IP address or DB2 communication port on the standby in step 1.c on page 226, change it back now. We changed our port number, so we change it back using the following commands:

```
db2 update dbm cfg using svcename 50000
db2stop force
db2start
```

f. Confirm that the database configuration parameters for the HADR standby are correct:

We expect four key HADR Primary database and Standby database configuration parameters of our example to be as shown in Table 7-2.

Table 7-2 Expected Primary and Standby HADR db cfg values

	Primary	Standby
HADR local host name (HADR_LOCAL_HOST)	LEPUS	MENSA
HADR local service name (HADR_LOCAL_SVC)	55001	55002
HADR remote service name (HADR_REMOTE_HOST)	MENSA	LEPUS
HADR remote service name (HADR_REMOTE_SVC)	55002	55001

In our experience, these values were already correct after the **restore database** on our standby server. If this was not the case, we would issue appropriate **db2 update db cfg for sample using** *<parm>* *<value>* commands and a **deactivate database** command to put them into effect.

- g. Start HADR on the standby (MENSA):  
db2 start hadr on database *dbname* as standby
- h. Log on to the primary server.
- i. Start HADR on the primary LEPUS:  
db2 start hadr on database *dbname* as primary
- j. Check HADR state until the connection succeeds and is approaching or at Peer state. See Chapter 4, “HADR administration and monitoring” on page 99 for the details of monitoring HADR. We expect to see something like the output in Example 7-4 from a **db2pd -d *sample* -hadr** command.

*Example 7-4 DB2 HADR in Connected, Peer state at V9.5*

---

```
db2inst1@lepus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days
00:00:15

HADR Information:
Role      State                SyncMode HeartBeatsMissed
LogGapRunAvg (bytes)
Primary Peer              Sync      0                    0

ConnectStatus ConnectTime                Timeout
Connected    Thu May 22 14:28:15 2008 (1211491695) 120

LocalHost                LocalService
lepus                     55101

RemoteHost                RemoteService
RemoteInstance            mensa                     55102
db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000015.LOG 0          0x0000000004A38000

StandByFile StandByPg StandByLSN
S0000014.LOG 0          0x0000000004650000
```

---

At this stage, the outages are over, all our work is complete and users can be allowed to reconnect to the HADR primary.



This concludes our description of a DB2 migration using command line techniques in an HADR configuration to minimize downtime in a production environment.

## 7.3 Rolling OS/Application/DB2 configuration parameters

This section covers high level instructions for updating database and database manager configuration parameters, as well as for any operating system (OS) and application upgrades. In order to keep your HADR systems optimal, you should apply changes to both servers as quickly as possible.

Note that when updating HADR-specific database configuration parameters, DB2 will not allow you to switch HADR roles. Certain HADR parameters must always remain matched, such as `HADR_TIMEOUT` for example. This means you will require a short database outage while updating the parameters on the primary server and recycling the DB2 instance or deactivating/activating the database.

For the HADR Primary, this process of role switching is not required for those databases or database manager configuration parameters which are dynamic - that is, which require no recycling of the DB2 database or instance to take immediate effect. The list of database and database manager configuration parameters showing which are dynamic, as denoted by the column heading "Cfg Online", can be found in the DB2 V9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.config.doc/doc/r0005181.html>

Having said that, after the dynamic database manager or database configuration parameter update has been issued on the HADR primary, it should be manually performed against the HADR standby as well, because parameter configurations are not logged and thus not mirrored through HADR log shipping.

## Procedure

In our procedure we use the example environment of Server A and B. The DB2 database on Server A is initially the HADR primary and B is the standby.

1. On Server B in HADR Standby role:
  - a. Make sure that your HADR is in a Peer state with db2pd, snapshot, or HADR manage window.
  - b. Issue the **deactivate database** command.
  - c. If necessary, stop the DB2 Instance. This applies to Operating System or non-DB2 changes which require a server recycle, in addition to Database Manager Configuration Parameters which require a DB2 Instance recycle.
  - d. Make the necessary changes to the hardware, software or DB2 configuration parameters.
  - e. Start the DB2 Instance if it was stopped.
  - f. Issue the **activate database** command, or explicitly connect to the database.
  - g. Check HADR to ensure Peer state by db2pd, snapshot, or HADR manage window.
  - h. Switch roles of the primary and standby by issuing an unforced **takeover hadr** command against the standby database.
  - i. Redirect clients to the HADR primary database on server B. This can be done by Automatic Client Reroute (ACR), or an IP address switch.
2. On Server A in HADR Standby role:
  - a. Repeat steps 1 a to i, but run against Server A.

That concludes our exploration of rolling DB2 fix pack applies with no downtime, as well as DB2 migrations and non-DB2 maintenance work with minimal downtime, all made possible with HADR and some effective techniques.



## DB2 with TSA

In this chapter we explain how to integrate DB2 in a Tivoli System Automation (TSA) environment. We provide the basic management concepts with recommendations and considerations to reduce the time consumed for failover. The configuration procedures are demonstrated with DB2 9.1. For DB2 9.5, the db2haicu utility provides an easy way to configure the cluster environment with DB2 in a shared disk.

We cover the following topics:

- ▶ Overview
- ▶ How DB2 works with TSA
- ▶ Planning a TSA cluster
- ▶ Setting up a TSA cluster
- ▶ Considerations for the db2nodes.cfg file
- ▶ DB2 9.5 High Availability Feature

## 8.1 Overview

IBM Tivoli System Automation (TSA) for Multiplatforms provides a framework to manage automatically, the availability of components known as resources. Some examples of resources include:

- ▶ Any piece of software for which start, monitor, and stop scripts can be written to control.
- ▶ Any Network Interface Card (NIC) to which TSA has been granted access. That is, TSA will manage the availability of any IP address that a user wants by floating that IP address among NICs to which it has been granted access.

For example, both a DB2 instance, and the file systems that are used by DB2, have start, stop, and monitor commands. Therefore, TSA scripts can be written to manage these resources automatically. Scripts, as well as other attributes of a resource, are required by TSA to manage that resource. TSA stores a resource's attributes in an object container, much like the attributes of a Java class. In fact, TSA manages a resource by instantiating a class for that resource.

TSA also allows related resources to be managed in what are known as resource groups. TSA guarantees that all resources within a given resource group are online at one and only one physical node at any point in time. Also, all of those resources will reside on the same physical node. Examples of resource groups (such as related resources) are a DB2 instance, its IP address, and all of the databases that it manages.

Finally, TSA provides high availability (HA) for any resource group that it manages, by restarting all of its resources if it fails. The resource group is restarted on an appropriate node in the currently online cluster domain. An appropriate node must contain a copy of all of the resources, which are defined in the failing resource group, to be selected as a node to restart on.

### 8.1.1 TSA components

Since V2.1, TSA consists of these components:

- ▶ *End-to-end automation component:*

This is the second-tier-enabling technology, which gives TSA a rare ability to provide control over heterogeneous environments of multiple base-level clusters. The low level clustering software, currently installed, is not replaced. TSA end-to-end automation can control an existing HACMP on AIX, Heartbeat on Linux, or MSCS just as easily as if it were controlling a TSA base component cluster on these same platforms. TSA software interfaces

called *End-to-End Automation Adapters* operate between TSA and the base-level clustering software.

For more details, refer to:

*End-to-end Automation with IBM Tivoli System Automation for Multiplatforms, SG24-7117.*

► *Base component:*

This TSA component provides base-level cluster functionality. It acts as an equivalent to HACMP or other clustering software, on AIX and Linux. The TSA base component consists of the following parts:

a. *Automation adapter:*

At the base component level, a *first level automation adapter* is used to directly control resources on nodes inside the base-level cluster.

At the end-to-end automation component level, an *end-to-end automation adapter* provides an interface between the automation layer already existing within a single base-level cluster, and the end-to-end automation layer which spans multiple clusters.

b. *Operations console of the base component:*

The TSA Operations Console is a Web browser accessed feature (no client software install required), based on the WebSphere Portal Server Integrated Solutions Console. Because it spans all TSA domains, and provides the same look and feel, there is no interface discrepancy between platforms.

While the Operations Console is there to help monitor the automation and current states of various resources, it also allows manual resource state changes (for example, if a server is required to be taken offline for maintenance), and performs takeovers accordingly. The Operations Console can operate in three different modes:

- i. *End-to-end automation mode:* Where end-to-end automation is essentially active.
- i. *First-level automation mode:* Where end-to-end automation has been installed, but is not active.
- ii. *Direct access mode:* Where you are essentially running the console directly on a system that is part of a base-level cluster. TSA end-to-end automation itself is not generally installed here. There is just an automation adapter plug-in on base-level cluster nodes for interfacing to non-TSA clustering software.

For more details, refer to:

*Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide, SC33-8272.*

## 8.1.2 Terminology of TSA

The following terms are common between TSA and other clustering solutions:

- ▶ *Resource*: This can be a physical device such as: a disk, a network adapter; or software, from the Operating System itself, through middle ware and DBMS, to WebSphere Administration Server (WAS) and Web Server daemons. With respect to up/down status monitoring and automation (and now end-to-end automation), resources can also exist at multiple tiers, for example, grouping granular resources and base-level cluster environments to make the concept of an application environment.
- ▶ *Critical resource*: This is a resource that must not be active on more than one node at the same time. A primary example of this is a static IP address that is shared between the nodes which clients use to connect to server resources. By this definition, the HADR primary role database would also be a critical resource.
- ▶ *Cluster/peer domain*: This is a group of host systems or nodes. These are what resources are part of, so if groups of resources become unavailable, it essentially indicates that a given node, or subcluster is unavailable. This leads to automated actions designed to maintain availability by establishing *quorum*, and to attempt restoration of the failed nodes without conflicting with resources on available nodes.
- ▶ *Quorum*: The majority level of operational nodes, required before certain operations, can be performed within a cluster. TSA splits this definition into:
  - *Configuration quorum*: Determines whether cluster configuration changes are accepted.
  - *Operational quorum*: Determines whether state changes of resources are accepted (so as not to cause conflicts).
- ▶ *Tiebreaker*: Where equal number of nodes exist in a subcluster, this determines which of those subclusters can have the quorum. The mechanism of the tiebreaker can be of six different types, depending on the platform and devices available:
  - *Operator*: Requires human intervention for a decision in order to reach a quorum.
  - *Fail*: A default/pseudo tie-breaker in which neither subcluster attains quorum.

- *SCSI*: Shared SCSI disk implementation on certain Linux platforms. SCSI reserve or persistent reserve command is used to achieve tie-breaker exclusive status.
- *ECKD™*: Shared ECKD disk on system z Linux. The ECKD reserve command is used to achieve tie-breaker exclusive status.
- *Disk*: Shared SCSI-like disk on AIX. An equivalent or pseudo-SCSI reserve or persistent reserve command is used to achieve tie-breaker exclusive status on SCSI-like or emulated disk.
- *EXEC*: Network implementation that calls a custom executable to achieve a Reliable Scalable Cluster Technology (RSCT) exec tie-breaker. Internet Control Message Protocol (ICMP) ping echo requests are sent to an external IP instance to determine tie-breaker exclusive status in the case of subcluster nodes failing. Note that this method relies on failure of ICMP response to indicate subcluster/node failure and is unable to determine if communication is still working between the subcluster peer nodes.

In the terminology specific to TSA, the hierarchy by which Tivoli System Automation coordinates resources in a clustered environment comprises an *Automation Manager*, which consists of a *Binder* and a *Logic Deck*.

- ▶ *Binder*: Has resources assigned (bound) to a particular node. If the resource cannot be accessed on any node, then it is placed in a *sacrificed* state. If it can be accessed, then it is placed in a *bound* state to that node. Resources in a bound state are only actually started if resource dependencies have been met.

If System Automation has not yet tried to access a given resource, it is by default in an *unbound* state. Conflicts between multiple available resources are managed by a priority value. The resource with lower priority is placed in *sacrificed* state.

- ▶ *Logic deck*: Sends orders to start and stop resources according to the rules set out by the *Automation Policy*.
- ▶ *Automation Policies*: An abstract business rule-based set of dependencies and relationships between resources, rather than hard and fast scripts. The Automation Policies determine what should occur when values change in the monitored states of resources, specifically when differences occur between a *desired state*, (Online/Offline), and an *observed state*.
- ▶ *Resource relationships*: A critical part of the automation policy:
  - *Stop-start relationships* determine which resources should be stopped or started in a given scenario.
  - *Location relationships* determine whether related resources should be started on the same or on separate nodes.

- ▶ *Equivalency*: A group of resources that can be said to provide the same function: TSA can choose from any one of these in order to provide a given function in the required definition of availability. For example, given a number of redundant network adapters, if one fails, that IP address can be mapped by TSA to another adapter in the equivalency.

Understanding the interaction between the components within this hierarchy is beyond the scope of this book. The reference material for TSA is as follows:

- ▶ *Tivoli System Automation for Multiplatforms Version 2.1, program number 5724-M00 - Base Component User's Guide*, SC33-8210-05.
- ▶ *Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide*, SC33-8272.
- ▶ *Tivoli System Automation for Multiplatforms Version 2.2 - Installation and Configuration Guide*, SC33-8273-00.
- ▶ *Tivoli System Automation for Multiplatforms Version 2.2 - End-to-End Automation Management Component Administrator's and User's Guide*, SC33-8275-00.
- ▶ *End-to-end Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117.
- ▶ Tivoli System Automation V2.3 manuals can be found at the Web site:  
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.3.html>
- ▶ Tivoli System Automation V2.2 manuals can be found at the Web site:  
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>
- ▶ This is the TSA introduction Web page:  
<http://www.ibm.com/software/tivoli/products/sys-auto-linux/index.html>

## 8.2 How DB2 works with TSA

Technically, TSA Automation Policies themselves do not entail scripting. These policies are in the form of business rules or heuristics that dictate, at a high level, how resources on various nodes should behave and interact. This enables attribution of priorities such that for any given partial or total resource failure scenario, the appropriate action is taken.



## How TSA detects failures

TSA controls the DB2 instance and DB2 data objects resources as application resources. For the application resources, TSA detects the failure by using the monitor scripts. When you create the application resource, you have to register the monitor script to the application resource with a valid argument to detect the failure with a proper return code. Refer to “Create the storage resources” on page 263 for the syntax of creating a resource.

The monitor script is executed by TSA regularly and returns the return code (RC) to TSA. TSA decides on the next action based on this return code. Table 8-1 and Figure 8-2 list the meanings of the return codes and the actions that TSA will take when the nominal state is online. If RC=1, this resource is recognized to be “Online,” and TSA takes no action. If the monitor script returns a “2” RC, this resource is considered to be “Offline,” and TSA will first stop the related resources, then start this resource again.

Table 8-1 Return codes from the monitor script and actions taken by TSA

RC	OpState	Meaning	First action of TSA (nominal state=online)	Second action of TSA (nominal state=online)
0	Unknown <sup>a</sup>	Unknown (monitor script timed out)	Kill the monitor script	Nothing
1	Online	Running	Nothing	Nothing
2	Offline	Not running	After stopping the related resources, start the resource. <sup>b</sup>	Nothing
3	Failed Offline	Failed (cannot recover)	Stop the related resources	Perform takeover <sup>c</sup>
4	Stuck Online	Start or stop command times out. Need operation by user	Nothing	Nothing
5	Pending Online	Now starting	Nothing	Nothing
6	Pending Offline	Now stopping	Nothing	Nothing

a. The script cannot use return code 0; this return code is used by TSA when the monitor script timeout occurs.

b. If there are any related resources to the failed resource, TSA stops the related resources in advance.

c. The Mandatory attribute value of this resource should be true.

Table 8-2 Return code from the start/stop script and TSA's actions

RC	Meaning	First action of TSA (nominal state=online)	Second action of TSA (nominal state=online)
0	Start/stop command completed successfully	Move next start/stop sequence	Nothing
not 0	Start command failed (Failed Offline)	Stop the resource	Perform takeover (see note3)
not 0	Stop command failed	Nothing	Nothing

Figure 8-1 illustrates the typical behavior when a resource in one node failed. Note that the return codes shown in the figure are from the start script. The return codes from the monitor script are not shown. When the monitor script detects a resource failed on node1, it returns RC=2 (offline). TSA executes the start script. After this resource is started, the monitor script returns RC=1 (online) in the next check. When the resources failed again, the monitor script returns RC=2 (offline). TSA executes the start script. However, if the start script cannot start the resource, it returns RC=1, which triggers the failover action. TSA then tries to stop the resource on node1. Following the stop, TSA fails the resource groups on node1 over to node2.

This is a typical recovery behavior. The DB2 resource, the mount point resources, and LVM resource all are operated in this way.

For more information, refer to Chapter 7: How IBM Tivoli System Automation processes the system information of *Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide*, SC33-8272.

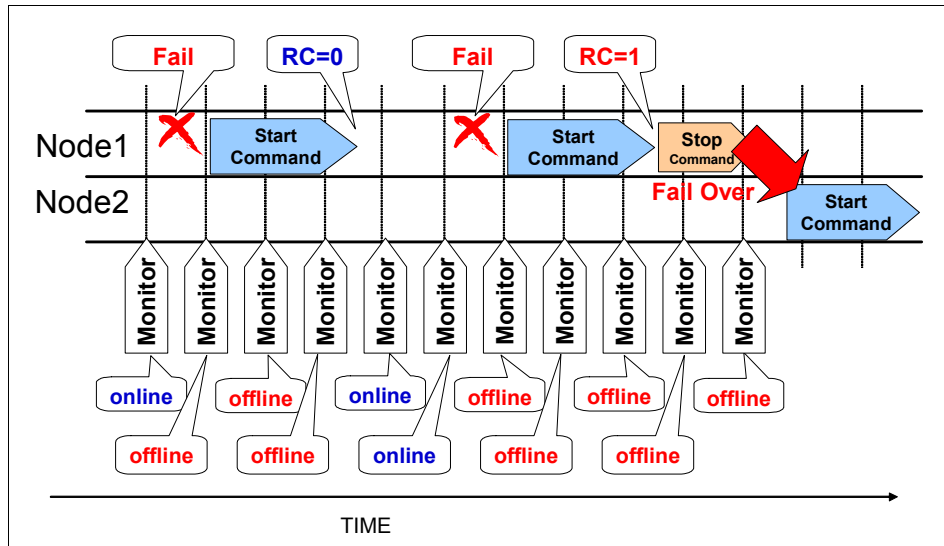


Figure 8-1 The failed resource and recovery

**Tips:** High availability failover behavior for TSA is also set up using configuration scripts. For those who like to have minimum scripts, a reference article for simplifying the configuration of failover can be found at the Web site: <http://www.ibm.com/developerworks/ibm/library/i-odoebp16/>

This article can be particularly valuable to those having a system environment similar to the test platform used in the article. This test platform has an application running on a remote Network File System (NFS) mount, IBM HTTP Web Server, J2EE, WebSphere Administration Server (WAS), and DB2.

## 8.3 Planning the high availability cluster

When you plan the high availability cluster, you can choose the configurations that best fit your system environment:

► Active/standby:

The typical configuration is an active/standby configuration, where one node is active and the other is on standby. One resource group is shared between the nodes. This configuration is very simple and easy to manage, but one node is not utilized for service.

► Active/active:

The active/active configuration is also referred to as mutual takeover. In an active/active configuration, a database is usually running under each node. Figure 8-2 illustrates the mutual takeover configuration. Database\_A is running on node\_A and database\_B is running on node\_B. If node\_A fails, the resource group for database\_A is taken over by node\_B. Both database\_A and database\_B are then running on one node: node\_B. If node\_B crashes, the same operation happens, just in the opposite direction.

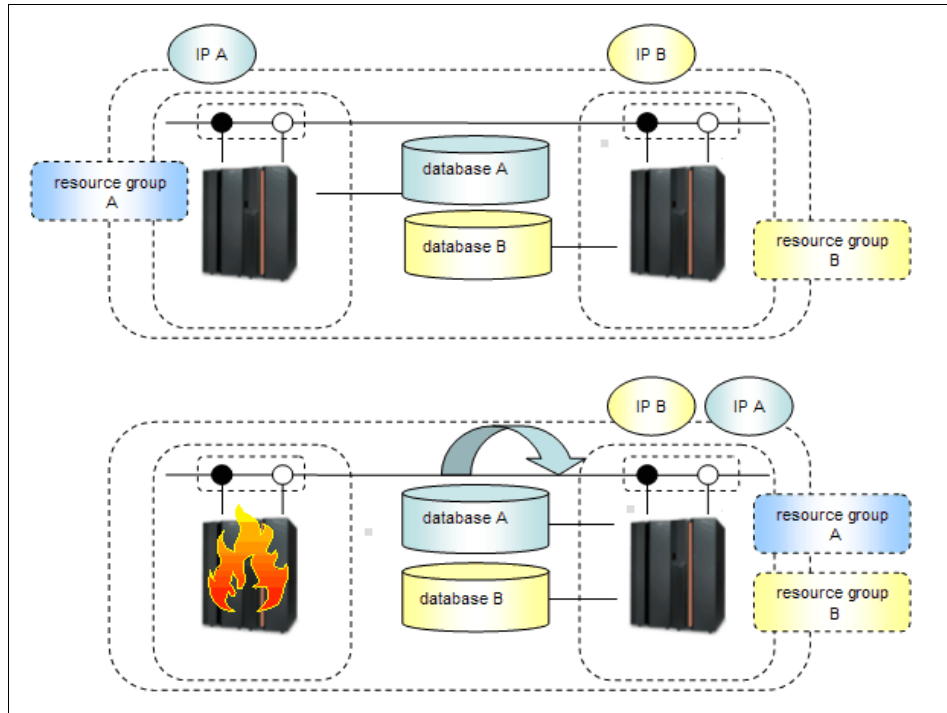


Figure 8-2 Mutual takeover

This configuration is called active/active because each node has its own database running and both are active at the same time. In the meantime, both nodes play the standby role for the other one and can take over each other's resources. That is why this configuration is also called a mutual takeover configuration. With this configuration, you can exploit both machine resources for services, but this is more complicated as compared to active/standby.

For this configuration, you need to plan carefully so that each node has enough machine resources (CPU or physical memory) to carry two databases while running. Let us say you have 1 GB physical memory on each node and have allocated 600 MB for a DB2 buffer pool in each database. After node\_A fails, database\_A is moved to node\_B. The total memory requirement now exceeds what node\_B has. What happens next is a huge page out to paging

space, which sometimes causes the system to hang; and at the very least, impacts performance to a degree where no work is possible.

► Other variations:

There are some variations of cluster configurations for more than three nodes in a cluster, as shown in Figure 8-3.

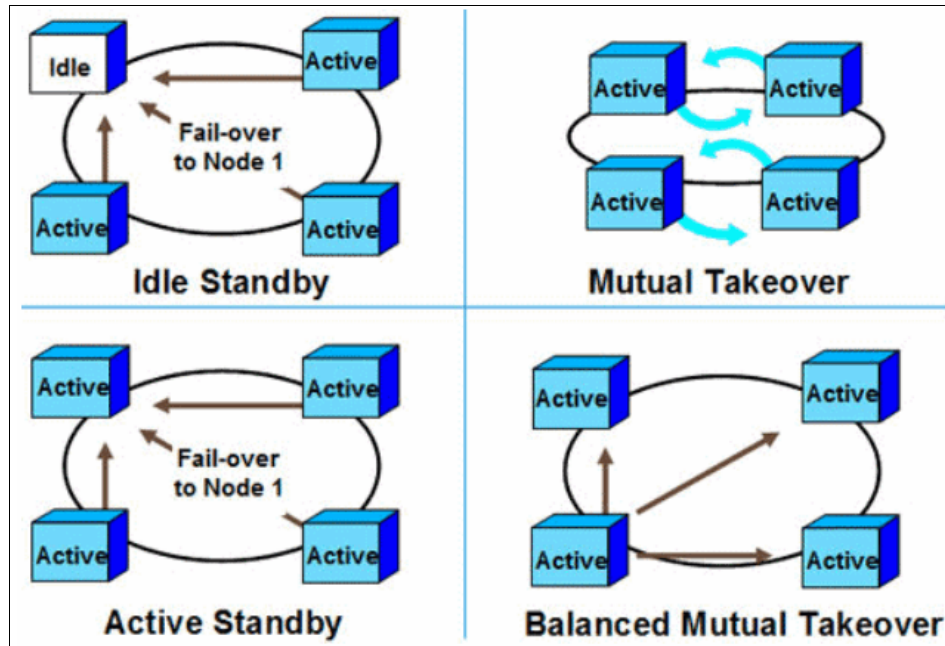


Figure 8-3 Variations of cluster configuration for multiple nodes

For more details about the DB2 highly available data store, refer to the IBM white paper “DB2 Universal Database and the Highly Available Data Store” at:

<http://www.ibm.com/developerworks/db2/library/techarticle/0310me1nyk/0310me1nyk.html>

## 8.4 Setting up TSA with DB2 9.1

In this section, we focus on the setup procedure for one Active/standby cluster with a shared disk. We start from the planning of the cluster domain, then proceed to the TSA installation and configuration.

Two main steps for setting up the system environment of a cluster domain are:

1. Planning the cluster domain

## 2. Configuration of TSA and DB2

### 8.4.1 Planning the cluster domain

We use the following resources to create the sample cluster domain:

- ▶ Two servers, node1 and node2, running AIX5.3 TL7 SP3-0811.
- ▶ An external storage that can be accessed from both servers.
- ▶ Both servers can communicate through two individual networks. This enables the cluster domain to have the redundant heartbeat paths.
- ▶ Each server can reach the third machine as a tiebreaker.
- ▶ DB2 Version 9.1 FP4a.
- ▶ Tivoli system automation for multiplatform Version 2.2

Figure 8-4 shows the architecture of this cluster domain. This is a simple cluster on two nodes. During normal operation, one server in the cluster domain owns all the resources to run the DB2 instance. We call this server the “owner node”. When the cluster software detects a failure on the owner node, the cluster software fails the resources over to the other server to continue DB2 service. The server that has taken over the resources is called the “failover node”.

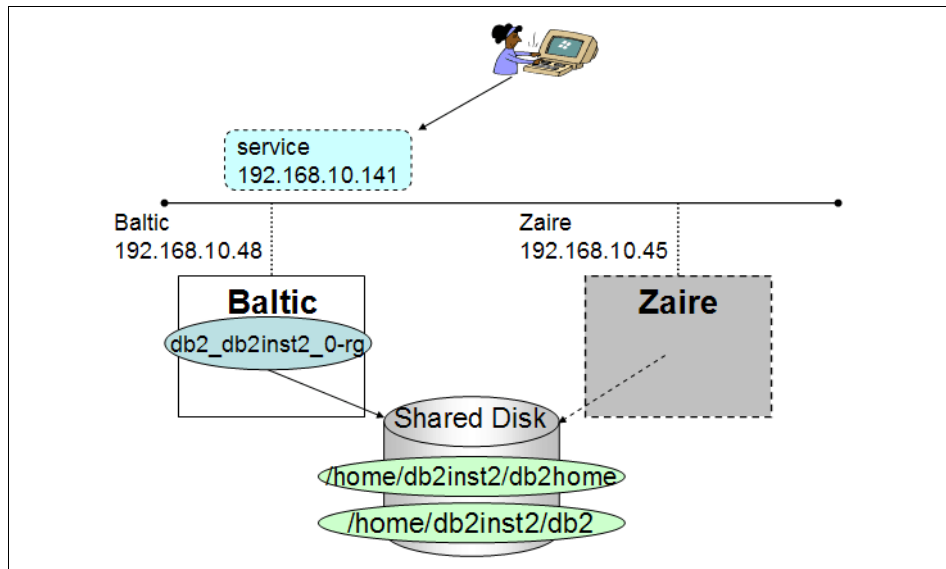


Figure 8-4 The architecture of the sample cluster domain

After the node role in the cluster domain is decided, we can plan the TSA configuration accordingly. The TSA configuration includes resource groups, resources, equivalencies, relationships, and scripts to control the application resources.

Figure 8-5 shows the relationship of the resource group and the resources. We configure the `db2_db2inst2_0_rg` resource group and define the resources listed in Table 8-3 as the members of this resource group.

*Table 8-3 Resources*

Resource	Description
<code>db2_db2inst2_0-rs</code>	DB2 instance
<code>db2mnt_db2inst2_db2home-rs</code>	DB2 instance directory mount point in shared disk
<code>db2mnt_db2inst2_db2-rs</code>	DB2 data directory mount point in shared disk
<code>db2_db2inst2_0-rs-ip</code>	Service IP address

In our cluster domain, when TSA starts up the DB2 instance, the mount points and LVM as well as the service IP address must have been started. Besides, TSA does not stop the DB2 instance until these three resources are stopped.

To meet these requirements, we define the following relationships between these resources:

- ▶ The DependsOn relationships between the `db2_db2inst2_0-rs` resource and the resources for the mount points.
- ▶ The DependsOn relationships between the resources for the mount points and the resource for LVM.
- ▶ The DependsOn relationships between the resource for LVM and the service IP address.

Usually an equivalency is used to define the policy for the selection of the network interfaces. In our environment, we define the equivalency that has equal priority between the `en1` of `node1` and the `en1` of `node2`. The DependsOn relationship is defined between the resource for service IP address and the network equivalency. When TSA detects a network interface failure with the service IP address, with this equivalency, the IP address takeover is done immediately:

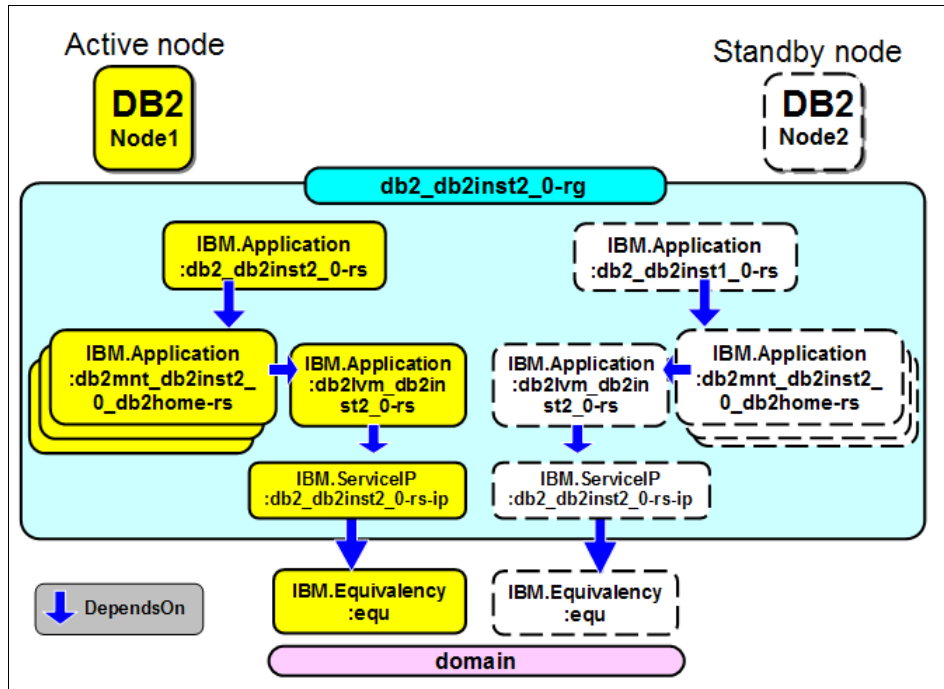


Figure 8-5 Resource relationships

## 8.4.2 Configuration of TSA and DB2

In this section we explain the configuration procedure for a shared disk HA cluster with DB2 V9.1 and TSA.

### Preparing the operating system environment

The first task is to prepare the operating system environment using these steps:

1. Check the prerequisites:

See these Web sites for the hardware and software requirements:

- The minimum software requirement for running DB2 on AIX can be found at the following Web site:

<http://ibm.com/software/data/db2/9/sysreqs.html>

- For information about software requirements for running TSA, refer to:

<http://ibm.com/software/tivoli/products/sys-auto-linux/platforms.html>

- Tivoli System Automation V2.2 manuals can be found at the Web site:



<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>

- RSCT for AIX manuals can be found at the Web site:

[http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct\\_aix5153/b15adm1110.html](http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct_aix5153/b15adm1110.html)

## 2. Set the environment variables:

We have to set two environment variables to use the TSA environment. The CT\_MANAGEMENT\_SCOPE environment variable must be set for all users of TSA on all nodes. Add these two entries to /etc/profile:

```
export CT_MANAGEMENT_SCOPE=2
```

For Linux environment, add to /etc/profile.local.

Verify the environment variable settings as shown in Example 8-1.

*Example 8-1 Checking value of environment variables*

---

```
(0) (F) root@node1 # cat /etc/profile
..
# add for TSA/DB2 HA Configuration
export CT_MANAGEMENT_SCOPE=2
(0) (F) root@node1 # echo $CT_MANAGEMENT_SCOPE
2
```

---

## 3. Edit the /etc/hosts files:

We assign the following static IP addresses to the en0 adapters on the owner and failover nodes:

```
Owner node (Node1)
en0: 9.188.198.119 (255.255.255.0)
Failover node (Node2)
en0: 9.188.198.118 (255.255.255.0)
```

All cluster nodes must have static IP addresses. Make sure that the /etc/hosts file of every cluster node contains entries of the owner node and the failover node names. Example 8-2 shows the entries in the /etc/hosts file of both servers:

*Example 8-2 The entries in /etc/hosts on node1 and node2*

---

```
9.188.198.119 node1
9.188.198.118 node2
```

---

Two servers have to be able to communicate to each other through a public network. We confirm it by a **ping** command. Run the following commands on both nodes and make sure that they complete successfully:

```
(0) (F) $ ping node1
(0) (F) $ ping node2
```

4. Confirm the broadcast addresses:

TSA uses the broadcast address to check the heartbeat. An incorrect broadcast address can cause unexpected failover. Usually a broadcast address is set automatically, so verify that the configuration of the broadcast address is correct as shown in Example 8-3.

*Example 8-3 Checking the broadcast address*

---

```
(0) (F) root@node1 # ifconfig -a
en0:
flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD(ACTIVE),PSEG,LARGESEND,CHAIN>
    inet 9.188.198.119 netmask 0xffffffff broadcast 9.188.198.255
    tcp_sendspace 131072 tcp_recvspace 65536
en1:
flags=5e080863,c0<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GROUPRT,64BIT,CHECKSUM_OFFLOAD(ACTIVE),PSEG,LARGESEND,CHAIN>
    inet 192.168.0.15 netmask 0xffffffff broadcast 192.168.0.255
    tcp_sendspace 131072 tcp_recvspace 65536
```

---

You can calculate the correct broadcast address from the IP address and the netmask. The broadcast address is computed by bitwise OR operation between IP address and the reversed netmask. The reversed netmask is the reverse bit sequence of the netmask. For example, the reversed netmask of “255.255.255.0” is “0.0.0.255”. Table 8-4 shows a sample calculation of the broadcast address.

*Table 8-4 The calculation of the broadcast address*

	IP address	IP address by binary
IP Address	9.188.198.119	00001001.10111100.11000110.01110111
Reversed netmask	0.0.0.255	00000000.00000000.00000000.11111111
Broadcast address	9.188.198.255	00001001.10111100.11000110.11111111

If the broadcast address is not correct, you can change it by **smit**.

- a. Start up the **smit** menu by the **smitty tcpip** command.

- b. Select **Further Configuration** → **Network Interfaces** → **Network Interface Selection** → **Change / Show Characteristics of a Network Interface**
- c. Select the network interface whose configuration you want to change.  
If your platform is Linux, you can use the **ip** command to do so.

## Setting up the storage for sharing

Before proceeding with the steps to set up the storage for sharing between two nodes, the storage should be configured and the volume groups created. Each node of the cluster domain has to be able to access the shared volume groups. The owner node keeps the ownership of the shared volume groups during the normal operation. The failover node takes over the ownership and varies on the volume groups automatically in case of a takeover. Therefore, make sure that both servers can vary on the volume groups.

For the details about storage configuration, refer to the documents provided by your storage vendors, for example, *IBM System Storage DS4000 Series and Storage Manager, SG24-7010*.

The shared storage setup steps are as follows:

1. Check the volume group major numbers on the owner node:

Ensure that the major numbers of all volume groups on the failover node are the same as the ones on the owner node. You can use the following command to obtain the volume group major number:

```
(0) # ls -al /dev/<volume group name>
```

The volume group major number is equivalent to the major device number of the special device file. Example 8-4 shows a command sample and output. In this example, the output shows that the volume group major number of “sharevg” is 48. Repeat these steps for all the volume groups.

*Example 8-4 Sample output of special device file*

---

```
(0) # ls -la /dev/sharevg
crw-rw---- 1 root system 48, 0 Apr 14 19:29 /dev/sharevg
```

---

2. Determine if the same major number is available on the failover node:

Execute the following command to determine the available major numbers:

```
(F) # lvs -lsmajor
```

Example 8-5 shows the output of the **lvs -lsmajor** command. This output means that the failover node can use the volume group major number “48” for “sharevg”.

*Example 8-5 Sample output of the `lvfstmajor` command*

---

```
(F) #1v1stmajor
46..50,52..60,62...
```

---

3. Import the volume group from the owner node to the failover node:

If the major number is available on the failover node, import the volume group, specifying the required major numbers as follows:

```
(F) #importvg -y <volume group name> -V <first major number>
<pv name>
```

Example 8-6 shows how to import the volume group `sharevg`.

*Example 8-6 Sample output of the `importvg` command*

---

```
(F) # lspv |grep hdisk5
hdisk5          0009cdaa1628f8eb      None
(F) # importvg -y sharevg -V 48 hdisk5
sharevg
(F) # lspv |grep hdisk5
hdisk5          0009cdaa1628f8eb      sharevg          active
```

---

4. Create logical volumes:

In this scenario, we create two logical volumes on volume group `share VG` for the DB2 instance. One logical volume is for the DB2 home directory and the other one is for the database objects.

Example 8-7 shows the commands used in our environment.

*Example 8-7 Creating logical volume and file system commands*

---

```
(0) ### Create logical volumes on sharevg
mklv -t jfs2 -y lvdb2inst2 sharevg 10 hdisk5
mklv -t jfs2 -y lvdb2inst2db2 sharevg 10 hdisk5

(0) ### Create file systems on created logical volume
crfs -v jfs2 -d lvdb2inst2 -m /home/db2inst2/db2home -A no -a logname=INLINE
crfs -v jfs2 -d lvdb2inst2db2 -m /home/db2inst2/db2 -A no -a logname=INLINE

(0) ### Mount new file systems
mount /home/db2inst2/db2
mount /home/db2inst2/db2home

(0) ### Change owner
chown -R db2inst2:db2iadm1 /home/db2inst2/db2
chown -R db2inst2:db2iadm1 /home/db2inst2/db2home
```

---

5. Refresh the device files on the failover node:

After creating the logical volumes on the owner node, we have to refresh the volume group's device files on the failover node. The device files are special files stored in the /dev directory and are used for device control. Use the exportvg and importvg command to refresh these device files:

```
(F) # exportvg <volume group name>
(F) # importvg -y <volume group name> -V <first major number>
<pv name>
```

Example 8-8 illustrates that we first export volume groups by the exportvg command, then import the volume groups. The importvg command maintains the device files and the file entries in /etc/filesystems. After running the importvg command, check the device files and /etc/filesystems.

*Example 8-8 Refreshing the device files using exportvg and importvg*

---

```
(F) # varyoffvg sharevg
(F) # exportvg sharevg
(F) # importvg -y sharevg -V 48 hdisk5
sharevg
(F) # ls -l /dev/*db2inst2*
brw-rw---- 1 root    system    48, 3 Apr 22 15:19 /dev/lvdb2inst2db2
brw-rw---- 1 root    system    48, 2 Apr 22 15:19 /dev/lvdb2inst2home
crw-rw---- 1 root    system    48, 3 Apr 22 15:19 /dev/rlvdb2inst2db2
crw-rw---- 1 root    system    48, 2 Apr 22 15:19 /dev/rlvdb2inst2home
(F) # cat /etc/filesystems
...
/home/db2inst2/db2home:
    dev      = /dev/lvdb2inst2home
    vfs      = jfs2
    log      = INLINE
    mount    = false
    check    = false
    account  = false

/home/db2inst2/db2:
    dev      = /dev/lvdb2inst2db2
    vfs      = jfs2
    log      = INLINE
    mount    = false
    check    = false
    account  = false
```

---

**Note:** If you use row devices for a DMS table space container, you should check the access permission of the character device files after import. The **importvg** command resets the ownership of the device files to the original state “root:system”. If the DB2 instance ID should be the owner, you have to change them manually.

## Installing TSA and sample policies

In this section we explain how to install TSA and sample policies. We do not mention the installation of DB2. If you need the instruction for DB2 installation, refer to the DB2 information center. You have to install TSA and sample policies to both owner and failover node.

### *Installation of TSA*

Use the **installSAM** command in the CD or installation package to install TSA. You can install TSA for Multiplatform by using the following steps:

1. Execute the **installSAM** command.
2. Read the license agreement (scroll down to end of the file or press **q** twice.)
3. Enter **y** to accept the license agreement.
4. Check the installation result.

You can check the installation log to confirm that the installation has succeeded. Example 8-9 shows part of a message from a successful installation.

#### *Example 8-9 TSA installation result*

---

```
(0)(F) root@node1 # ./installSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on operating
system AIX 5300-06
installSAM: Packages will be installed from directory ./AIX
installSAM: The following package is not installed yet and needs to be installed
./AIX/sam.core

...

installSAM: The following license is installed
Product ID: 101
Creation date: Mon Mar 20 09:00:00 2006
Expiration date: Fri Jan 1 08:59:59 2038

installSAM: All packages were installed successfully
```

---

### *Install the sample policies*

Download the “Automation Policies” for your platform and install on both nodes from

<ftp://ftp.software.ibm.com/software/data/db2/express/linuxvalidate/db2salinux.tar.gz>

On AIX, use the **installp** command or **smit** to install the policies. On Linux, use **rpm** command. The installed sample policy files are located in the `/usr/sbin/rsct/sapolicies` directory. See Example 8-10.

*Example 8-10 Sample output of the /usr/sbin/rsct/sapolicies directory*

---

```
(0)(F) root@node1 # ls /usr/sbin/rsct/sapolicies
apache      itds        oracle      tsmclient
apache+data+ip lib         samba       tsmserver
bin         license.txt sap          tws
ccmdb       local_mount sendmail    was
glvm        maxdb       taddm
ihs         mqm         tec
ihs+data+ip nfsserver   tsmadminc
```

---

## Setting up the DB2 instance

The steps to create a DB2 instance on a shared disk are as follows:

1. Check the user ID and group ID of the DB2 instance owner on the nodes:

If there is no DB2 instance owner user, you have to create DB2 instance owner user and group first. The DB2 instance owner should have the same user ID and group ID on all the nodes in the cluster domain. We recommend that the DB2 instance owner have the same password on all cluster nodes.

Example 8-11 shows that the user ID of DB2 instance owner `db2inst2` is 205 and its group ID is 102. We have to ensure that they are same in every node.

*Example 8-11 Checking sample of user ID and group ID*

---

```
(0)(F) root@node1 # lsuser -a id pgrp db2inst2
db2inst2 ID=205 pgrp=db2iadm1
(0)(F) root@node1 # lsgroup -a ID db2iadm1
db2iadm1 ID=102
```

---

2. Create the DB2 instance:

We use `db2icrt` command to create DB2 instance. The command is in the instance directory on the DB2 install path.

```
(0) # <DB2 install path>/instance/db2icrt -u <fenced user name>
<instance owner name>
```

Example 8-12 and Example 8-13 illustrate how to create the DB2 instance and configure TCP/IP communication.

*Example 8-12 DB2 instance creation*

---

```
(0) root@node1 # cd /opt/IBM/db2/V9.1/FP4/instance
(0) root@node1 # ./db2icrt -u db2fenc1 db2inst2
DBI1070I Program db2icrt completed successfully.

(0) root@node1 # ./db2ilist
db2inst2
(0) root@node1 # su - db2inst2
(0) db2inst2@node1 $ db2start
04/21/2008 19:07:01    0  0  SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

---

*Example 8-13 Configuration of TCP/IP communication*

---

```
(0) db2inst2@node1 $ db2set db2comm=tcpip
(0) db2inst2@node1 $ db2 update dbm cfg using svcename db2_db2inst2
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
(0) db2inst2@node1 $ cat /etc/services |grep DB2_db2inst2
DB2_db2inst2      61001/tcp
DB2_db2inst2_1   61002/tcp
DB2_db2inst2_2   61003/tcp
DB2_db2inst2_END 61004/tcp
(0) db2inst2@node1 $ db2stop
04/21/2008 19:07:00    0  0  SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
(0) db2inst2@node1 $ db2start
04/21/2008 19:07:01    0  0  SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

---

3. Create the SAMPLE database:

We create the SAMPLE database to test the cluster domain on the /home/db2inst2/db2 directory (Example 8-14).

*Example 8-14 Create SAMPLE database*

---

```
(0) db2inst2@node1 $ db2 create database sample on /home/db2inst2/db2
DB20000I The CREATE DATABASE command completed successfully.
(0) db2inst2@node1 $ mkdir /home/db2inst2/db2/actlog
(0) db2inst2@node1 $ mkdir /home/db2inst2/db2/arclog
(0) db2inst2@node1 $ db2 update db cfg for sample using NEWLOGPATH
/home/db2inst2/db2/actlog
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
(0) db2inst2@node1 $ db2 update db cfg for sample using LOGARCHMETH1
disk:/home/db2inst2/db2/arclog
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
(0) db2inst2@node1 $ db2 backup db sample to /dev/null
```



Backup successful. The timestamp for this backup image is : 20080502052300

**(0) db2inst2@node1 \$ db2 connect to sample**

Database Connection Information

```
Database server      = DB2/AIX64 9.1.4
SQL authorization ID = DB2INST2
Local database alias = SAMPLE
```

---

**Note:** If you want to configure a shared disk HA on an existing DB2 instance, you have to move DB2 objects to the shared volume group. DB2 objects include the instance home directory, diagpath, and audit path as well as all the database objects.

4. Edit the db2nodes.cfg file:

The db2nodes.cfg file should contain the host name that is bound to the local IP address. When the failover node takes over the DB2 instance, you have to change the host name in the db2nodes.cfg file or have DB2 change the file by starting the DB2 instance with **db2gcf**. Refer to 8.5, “Considerations for the db2nodes.cfg file” on page 272 for more information. We choose “Using an alias in the hosts file” from the options introduced that section.

We specify an alias “db2host” as the host name in the /etc/hosts file on both nodes. See Example 8-15 and Example 8-16. This “db2host” host name always points to itself on each node. For example, if node1 tries to resolve the hostname db2host, the IP address resolved is 9.188.198.119.

*Example 8-15 The entries in /etc/hosts on node1*

---

```
9.188.198.119 node1 db2host
9.188.198.118 node2
```

---

*Example 8-16 The entries in /etc/hosts on node2*

---

```
9.188.198.119 node1
9.188.198.118 node2 db2host
```

---

We set this host name in the db2nodes.cfg file so we do not have to change the entry of this file when the DB2 instance is started on the other node.

Example 8-17 on page 256 shows how to change the db2nodes.cfg file. Edit the db2nodes.cfg file by the **vi** editor and change the hostname to “db2host”. This is not a required change. But making this change allows you to start the DB2 instance with the **db2start** command after a failover.

*Example 8-17 Changing steps of db2nodes.cfg file*

---

```
(0) db2inst2@node1 # db2stop
2008-05-02 01:41:57      0  0  SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.

(0) db2inst2@node1 # vi sql1lib/db2nodes.cfg
(0) db2inst2@node1 # cat sql1lib/db2nodes.cfg
0 db2host 0 db2host

(0) db2inst2@node1 # db2start
2008-05-02 01:42:03      0  0  SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
```

---

5. Edit the /etc/services file:

The SVCENAME database manager configuration parameter defines the TCP/IP service name of the service port number to which the DB2 instance listens. The /etc/services file on both nodes should have same DB2 service port name entries in order to run the DB2 instance correctly. The db2icrt command adds the entries to /etc/services file on owner node automatically, but you have to add them on the failover node manually. Example 8-18 shows the sample of node1 and node2.

*Example 8-18 The entries of /etc/services file on owner and failover nodes*

---

```
(0) root@node1 # cat /etc/services |grep db2inst2
DB2_db2inst2      61001/tcp
DB2_db2inst2_1   61002/tcp
DB2_db2inst2_2   61003/tcp
DB2_db2inst2_END 61004/tcp
(F) root@node2 # cat /etc/services |grep db2inst2
DB2_db2inst2      61001/tcp
DB2_db2inst2_1   61002/tcp
DB2_db2inst2_2   61003/tcp
DB2_db2inst2_END 61004/tcp
```

---

## Script preparation

In this section we discuss the scripts to be registered for TSA. They are used by TSA to control the application resources such as the DB2 instance and the mount points in the cluster environment.

### **Choose the base script**

The sample policies installed come with TSA scripts. We use the *sa-mount* script to control the mount point resources and the *sa-lvm* script to control the LVM resource. These scripts can be used for three types of action: stop, start, and monitor. The first argument specifies action. The syntax is as follows:

```
sa-mount [start|stop|status] <mount point>
sa-lvm [start|stop|status] <VG name> <PV name>
```

Table 8-5 lists the mount point resources for our scenario and the commands used to register these resources for the specific action. These commands are run in the step that creates the TSA resources.

*Table 8-5 Mount point resources and the corresponding register commands*

Resource	Action	Command for registering the resource
/home/db2inst2/db2home	Start	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount start /home/db2inst2/db2home</code>
/home/db2inst2/db2home	Stop	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount stop /home/db2inst2/db2home</code>
/home/db2inst2/db2home	Monitor	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount status /home/db2inst2/db2home</code>
/home/db2inst2/db2	Start	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount start /home/db2inst2/db2</code>
/home/db2inst2/db2	Stop	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount stop /home/db2inst2/db2</code>
/home/db2inst2/db2	Monitor	<code>/usr/sbin/rsct/sapolicies/bin/sa-mount status /home/db2inst2/db2</code>
sharevg (LVM)	Start	<code>/usr/sbin/rsct/sapolicies/bin/sa-lvm start sharevg hdisk5</code>
sharevg (LVM)	Stop	<code>/usr/sbin/rsct/sapolicies/bin/sa-lvm stop sharevg hdisk5</code>
sharevg (LVM)	Monitor	<code>/usr/sbin/rsct/sapolicies/bin/sa-lvm status sharevg hdisk5</code>

**Note:** In our scenario, we use this script without any modification. Because the system environment differs, you might need to modify the script to meet your requirements.

For the DB2 instance, we use the TSA scripts provided by DB2 in DB2 9.1 and later. These scripts are located in the `<DB2 install path>/ha/tsa` directory.

To consolidate the resource control scripts, we copy these scripts to our script directory `/usr/sbin/rsct/sapolicies/db2`. Example 8-19 lists the scripts copied.

Example 8-19 Copy the scripts to the /usr/sbin/rsct/sapolicies/db2 directory

```
(0) (F) root@node1 # cd /opt/IBM/db2/V9.1/FP4/ha/tsa
(0) (F) root@node1 # ls db2*
db2_monitor.ksh db2_start.ksh db2_stop.ksh
(0) (F) root@node1 # mkdir /usr/sbin/rsct/sapolicies/db2
(0) (F) root@node1 # cp -p * /usr/sbin/rsct/sapolicies/db2
(0) (F) root@node1 # ls /usr/sbin/rsct/sapolicies/db2/db2*
/usr/sbin/rsct/sapolicies/db2/db2_monitor.ksh
/usr/sbin/rsct/sapolicies/db2/db2_start.ksh
/usr/sbin/rsct/sapolicies/db2/db2_stop.ksh
```

Table 8-6 lists the actions and the commands used to register the resource db2inst2 for the specific action.

Table 8-6 Actions and registering command for the resource db2inst2

Resource	Action	Command for registering the resource
db2inst2	Start	/usr/sbin/rsct/sapolicies/db2/db2_start.ksh db2inst2 0
db2inst2	Stop	/usr/sbin/rsct/sapolicies/db2/db2_stop.ksh db2inst2 0
db2inst2	Monitor	/usr/sbin/rsct/sapolicies/db2/db2_monitor.ksh db2inst2 0

### Verify the scripts

We highly recommend confirming that these scripts will control your resources as you intend before starting the cluster domain creation. TSA detects the resource status from these scripts. If these scripts return the wrong status, it can cause unexpected failover of your cluster domain.

Here are the some major checks for these scripts. Ensure that your scripts behave as follows:

- ▶ All the start scripts can start the resources with return code “0”.  
You have to start all the resources in a correct order. For example, we make sure that our start scripts have the following task sequence:
  - a. Start the LVM resource.
  - b. Start the mount point resource for /home/db2inst2/db2home.
  - c. Start the mount point resource for /home/db2inst2/db2.
  - d. Start the DB2 instance resource.
- ▶ All the stop scripts can stop the resources with return code “0”.  
When stopping the resources, the task sequence should be the reverse order as specified in the start script.
- ▶ All the monitor scripts return the correct return code.

A monitor script should return a correct return code. When the resource is stopped, the correct return code is “2”. When the resource is started, the correct code is “1”.

## Preparation for setting clustering

Three tasks on nodes are required before starting node clustering:

### 1. SSH configuration:

Many of the TSA commands for setup require SSH on all nodes. This section describes how to set up SSH for root user so that no password is required when commands are entered. For additional information, consult the following article:

<http://ibm.com/developerworks/db2/library/techarticle/dm-0506finnie/>

SSH setup steps are as follows:

a. Log on as the root and execute all the steps on all nodes.

b. If the `/.ssh` directory does not yet exist, create it. Ensure that `/.ssh` directory does *not* allow write access for group and other.

c. From the `/.ssh` directory, generate a public key/private key pair:

```
(0)(F) #ssh-keygen -t rsa
```

When prompted for input, press Enter. Make sure that you do not enter a passphrase.

d. To enable the new key pair for use with ssh, execute the following commands:

```
(0)(F) #mv /.ssh/id_rsa /.ssh/identity
(0)(F) #chmod 600 /.ssh/identity
(0)(F) #cat /.ssh/id_rsa.pub >> /.ssh/authorized_keys
(0)(F) #chmod 644 /.ssh/authorized_keys
(0)(F) #rm /.ssh/id_rsa.pub
```

e. Use the `ssh-keyscan` utility to gather the public host key for each host. Substitute the IP addresses in the commands with yours.

```
(0)(F) #ssh-keyscan -t rsa node1, 9.188.198.119, node1_int,
192.168.0.5 >> /.ssh/known_hosts
(0)(F) #ssh-keyscan -t rsa node2, 9.188.198.118, node2_int,
192.168.0.15 >> /.ssh/known_hosts
```

f. Test that passwordless ssh is now enabled. From all nodes, run the following command:

```
(0)(F) #ssh node1 date
```

Make sure that this command succeeds without prompting you for additional verification.

## 2. Preparing nodes:

Before creating the cluster domain, run the `preprnode` command on every node that is a part of the cluster domain. The command syntax is as follows:

```
(0) (F) # preprnode <node name> <node name>...
```

In our environment, as a root user, we issue the command shown in Example 8-20 on both nodes, node1 and node2. You do not have to run this command for every DB2 instance, only once per node. Ensure that this command completes without any error.

*Example 8-20 Sample command of preprnode command*

---

```
(0) root@node1 # preprnode node1 node2  
(F) root@node2 # preprnode node1 node2
```

---

## 3. Creating the netmon.cf file:

When Broadcast pings are enabled between all nodes in the cluster domain, you do not need the `netmon.cf` file. But if you have disabled the broadcast ping (it is recommended in z/Linux environments), you must create this file to supply TSA with the IP addresses used for the quorum device. This file must contain the IP addresses that TSA can ping from both nodes as shown in Example 8-21. The IP addresses you can choose are the default gateway, the other servers in your system, and so on. When the heartbeat stops, TSA pings these addresses to determine whether it is the network interface failure or the partner node failure. In Example 8-21, we select the IP addresses 9.188.198.1 and 9.188.198.2.

*Example 8-21 The sample configuration of netmon.cf*

---

```
(0) (F) root@node1 # cd /usr/sbin/cluster  
(0) (F) root@node1 # cat netmon.cf  
9.188.198.1  
9.188.198.2
```

---

For TSA V2.2, the `netmon.cf` file should be deployed at `/usr/sbin/cluster`. This directory is changed to `/var/cf/cfg` directory from TSA V2.3 and later.

## System clock synchronization

Synchronizing the date and time among nodes on the cluster is not required. However, we recommend doing this because synchronized clocks can make problem determination more straightforward. When analyzing the logs on different nodes, you are able to see the time sequence of the events without any adjustment. Note that you can use the network time protocol (NTP) for this purpose. Refer to your operating system documentation for more information on how to configure NTP for your system.

## Configuring the DB2 cluster

Before starting the DB2 cluster configuration, confirm the following settings:

- ▶ The volume group that is used for DB2 objects is not varied on, on both nodes.
- ▶ The volume group is not set up to vary on automatically when the machine starts on either node.
- ▶ The major numbers of the volume groups are the same on both nodes.

### **Create the peer domain**

Create and start the peer domain by using the following commands. Only one execution on any one node is sufficient. You must specify all the nodes that you want to contain in this cluster domain.

```
(0) # mkrpdomain <domain name> <node> <node>...
(0) # startrpdomain <domain name>
```

Example 8-22 shows a procedure for creating, starting, and checking the status.

#### *Example 8-22 Create and start the peer domain*

---

```
(0) root@node1 # mkrpdomain shared_disk_domain node1 node2
(0) root@node1 # lsrpdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
shared_disk_domain Offline 2.4.8.3 No 12347 12348
(0) root@node1 # startrpdomain shared_disk_domain
(0) root@node1 # lsrpdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
shared_disk_domain Online 2.4.8.3 No 12347 12348
(0) root@node1 # lsrpnode
Name OpState RSCTVersion
node2 Online 2.4.8.3
node1 Online 2.4.8.3
```

---

### **Change the configuration of the communication groups**

You can change the heartbeat configuration with the **chcomg** command as shown in Example 8-23. You can check the output of the **lscomg** command to see if the broadcast configuration of all communication groups have been disabled.

#### *Example 8-23 Disabling the broadcast pings*

---

```
(0) root@node1 # lscomg
Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName ...
CG1 4 1 1 Yes Yes
CG2 4 1 1 Yes Yes
(0) root@node1 # chcomg -x b CG1
(0) root@node1 # chcomg -x b CG2
```

```
(0) root@node1 # lscomg
```

```
Name Sensitivity Period Priority Broadcast SourceRouting NIMPathName ...
CG2 4 1 1 No Yes
CG1 4 1 1 No Yes
```

---

## Defining the quorum

We highly recommend that you use the disk quorum to protect your resources and data in a production environment. The disk quorum is useful in preventing the “split brain” situation that might destroy the data.

For configuration procedure, refer to Chapter 9. Protecting your resources. quorum support of *Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide, SC33-8272*.

### Create the service IP resource

Create a service IP address for the DB2 service. To create and check this resource, you can use the `mkrsrc` command and `lsrsrc` command. In our environment, we specify two nodes in `NodeNameList`. If your cluster domain has more nodes, you should specify all the nodes you want to configure. Here is the command syntax:

```
(0) # mkrsrc IBM.ServiceIP Name="<resource name of service IP address>"
NodeNameList="{ '<node name1>', '<node name2>' }" IPAddress=<service IP
address> NetMask=<net mask>
```

```
(0) lsrsrc -Ab IBM.ServiceIP
```

Example 8-24 shows that the service IP `db2_db2inst2_0-rs-ip` is created.

#### Example 8-24 Create the service IP address resource

---

```
(0) root@node1 # mkrsrc IBM.ServiceIP Name="db2_db2inst2_0-rs-ip"
NodeNameList="{ 'node1', 'node2' }" IPAddress=9.188.198.53 NetMask=255.255.255.0
```

```
(0) root@node1 # lsrsrc -Ab IBM.ServiceIP
```

```
Resource Persistent and Dynamic Attributes for IBM.ServiceIP
```

```
...
```

```
resource 3:
```

```
Name = "db2_db2inst2_0-rs-ip"
ResourceType = 1
AggregateResource = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x00000000"
IPAddress = "9.188.198.53"
NetMask = "255.255.255.0"
ProtectionMode = 1
ActivePeerDomain = "shared_disk_domain"
NodeNameList = {"node1", "node2"}
OpState = 2
```



...

---

### **Create the storage resources**

We consider the mount point resources and LVM resource as the application resources and create them in the IBM.Application class. Here is the command syntax:

```
(0) # mkrsrc IBM.Application Name="<resource name>" ProtectionMode=1
StartCommand="<command line for starting the resource>"
StopCommand="<command line for stopping the resource>"
MonitorCommand="<command line for monitoring the resource>"
MonitorCommandPeriod=<number in second>
MonitorCommandTimeout=<number in second>
StartCommandTimeout=<number in second>
StopCommandTimeout=<number in second>
UserName="root" NodeNameList="{ '<node name1>', '<node name2>' }"
```

Note that the command should be in one line. For formatting purposes here, we split the command in multiple lines.

We create two mount point resources and one LVM resource as shown in Example 8-25.

#### *Example 8-25 Create the mount point and LVM resources*

---

```
# create resource for /home/db2inst2/db2home mount point
mkrsrc IBM.Application
Name="db2mnt_db2inst2_0_db2home-rs"
ProtectionMode=1
StartCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount start /home/db2inst2/db2home"
StopCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount stop /home/db2inst2/db2home"
MonitorCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount status
/home/db2inst2/db2home"
MonitorCommandPeriod=10
MonitorCommandTimeout=29
StartCommandTimeout=330
StopCommandTimeout=600
UserName="root"
NodeNameList="{ 'node1', 'node2' }"

# create resource for /home/db2inst2/db2home mount point
mkrsrc IBM.Application
Name="db2mnt_db2inst2_0_db2-rs"
ProtectionMode=1
StartCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount start /home/db2inst2/db2 "
StopCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount stop /home/db2inst2/db2"
MonitorCommand="/usr/sbin/rsct/sapolicies/bin/sa-mount status /home/db2inst2/db2"
MonitorCommandPeriod=10
MonitorCommandTimeout=29
```

```

StartCommandTimeout=330
StopCommandTimeout=600
UserName="root"
NodeNameList="{ 'node1', 'node2' }"

# create resource for LVM
mkrsrc IBM.Application
Name="db2lvm_db2inst2_0-rs"
ProtectionMode=1
StartCommand="/usr/sbin/rsct/sapolicies/bin/sa-lvm start sharevg hdisk5"
StopCommand="/usr/sbin/rsct/sapolicies/bin/sa-lvm stop sharevg hdisk5"
MonitorCommand="/usr/sbin/rsct/sapolicies/bin/sa-lvm status sharevg hdisk5"
MonitorCommandPeriod=15
MonitorCommandTimeout=14
StartCommandTimeout=30
StopCommandTimeout=30
UserName="root"
NodeNameList="{ 'node1', 'node2' }"

```

---

### ***Create the DB2 instance resource***

You can use the same command syntax for creating storage resources to create the DB2 resource. Example 8-26 shows how we create the DB2 instance resource `db2_db2inst2_0-rs`. We have already copied scripts from the directory `<DB2 install path>/ha/tsa` to the directory `/usr/sbin/rsct/sapolicies/db2`.

#### *Example 8-26 Create the DB2 instance resource*

```

(0) root@node1 # mkrsrc IBM.Application \
Name="db2_db2inst2_0-rs"
StartCommand="/usr/sbin/rsct/sapolicies/db2/db2_start.ksh db2inst2 0"
StopCommand="/usr/sbin/rsct/sapolicies/db2/db2_stop.ksh db2inst2 0"
MonitorCommand="/usr/sbin/rsct/sapolicies/db2/db2_monitor.ksh db2inst2 0"
MonitorCommandPeriod=10
MonitorCommandTimeout=29
StartCommandTimeout=300
StopCommandTimeout=300
UserName="root"
NodeNameList="{ 'node1', 'node2' }"

```

---

After creating the resources, verify the resource configuration by using the `lsrsrc` command. Example 8-27 shows the `lsrsrc` output section relevant to the DB2 resource. One resource can have multiple entries in the output, fixed resource, and floating resource entries. Check the configuration of the floating resource that has the value "1" on `ResourceType`.

#### *Example 8-27 Sample output of lsrsrc command for IBM.Application*

```

(0) root@node1 # lsrsrc -Ab IBM.Application

```

```

Resource Persistent and Dynamic Attributes for IBM.Application
...
resource 3:
  Name = "db2_db2inst2_0-rs"
  ResourceType = 1
  AggregateResource = "0x3fff 0xffff 0x00000000 0x00000000 0x00000000 0x00000000"
  StartCommand = "/usr/sbin/rsct/sapolicies/db2/db2_start.ksh db2inst2 0"
  StopCommand = "/usr/sbin/rsct/sapolicies/db2/db2_stop.ksh db2inst2 0"
  MonitorCommand = "/usr/sbin/rsct/sapolicies/db2/db2_monitor.ksh db2inst2 0"
  MonitorCommandPeriod = 10
  MonitorCommandTimeout = 29
  StartCommandTimeout = 300
  StopCommandTimeout = 300
...
  NodeNameList = {"node1","node2"}
...

```

---

### **Create the resource group**

Create the resource group and add the resources to the resource group using the following commands:

```

(0) # mkrp <resource group>
(0) # addrgmbr -g <resource group> <resource class>:<resource> ...

```

You can add multiple resources in one command using this format:  
 <resource class>:<resource>.

The **lsrg** command lists all the members of the resource group with their status. See Example 8-28. Ensure that all the resources are offline. If any resources are not offline, there is probably something wrong in the resource configuration. Check the configuration using the **lsrsrc -Ab <resource class>** command as shown in Example 8-27.

#### *Example 8-28 Create the resource group and check it*

```

(0) root@node1 # mkrp db2_db2inst2_0-rg
(0) root@node1 # addrgmbr -g db2_db2inst2_0-rg
IBM.Application:db2mnt_db2inst2_0_db2home-rs
IBM.Application:db2mnt_db2inst2_0_db2-rs
IBM.Application:db2lvm_db2inst2_0-rs
IBM.Application:db2_db2inst2_0-rs
IBM.ServiceIP:db2_db2inst2_0-rs-ip
(0) root@node1 # lsrg -m
Displaying Member Resource information:
Class:Resource:Node[ManagedResource]      Mandatory MemberOf      OpState
...
IBM.ServiceIP:db2_db2inst2_0-rs-ip          True      db2_db2inst2_0-rg      Offline
IBM.Application:db2_db2inst2_0-rs          True      db2_db2inst2_0-rg      Offline
IBM.Application:db2mnt_db2inst2_0_db2home-rs True      db2_db2inst2_0-rg      Offline

```

```
IBM.Application:db2mnt_db2inst2_0_db2-rs      True      db2_db2inst2_0-rg  Offline
IBM.Application:db2lvm_db2inst2_0-rs        True      db2_db2inst2_0-rg  Offline
```

---

### **Create the equivalency**

Create the equivalency for the network interfaces with the following command:

```
(0) # mkequ -p 0 <equivalency name> IBM.NetworkInterface:<network
interface name>:<node name>,<network interface name>:<node name>
```

In our environment, we create one equivalency on the en0 network interface as shown in Example 8-29.

#### *Example 8-29 Create the network equivalency*

---

```
(0) root@node1 # mkequ -p 0 equ IBM.NetworkInterface:en0:node1,en0:node2
(0) root@node1 # lsequ -Ab
Displaying Equivalency information:
All Attributes
```

```
Equivalency 1:
Name = equ
MemberClass = IBM.NetworkInterface
Resource:Node[Membership] = {en0:node1,en0:node2}
SelectString = ""
SelectFromPolicy = ORDERED
MinimumNecessary = 1
Subscription = {}
ActivePeerDomain = shared_disk_domain
Resource:Node[ValidSelectResources] = {en0:node1,en0:node2}
Resource:Node[InvalidResources] = {}
ConfigValidity =
AutomationDetails[CompoundState] = Automation
```

---

### **Create the relationships**

Create the relationships to define the operational order. In our environment, we use the DependsOn relationship as shown in Example 8-30. Refer to 8.4.1, “Planning the cluster domain” on page 244 for more information.

#### *Example 8-30 Create relationships by mkrel command*

---

```
(0) root@node1 # mkrel -p DependsOn -S IBM.ServiceIP:db2_db2inst2_0-rs-ip -G
IBM.Equivalency:equ
(0) root@node1 # mkrel -p DependsOn -S
IBM.Application:db2mnt_db2inst2_0_db2home-rs -G
IBM.ServiceIP:db2_db2inst2_0-rs-ip
(0) root@node1 # mkrel -p DependsOn -S IBM.Application:db2mnt_db2inst2_0_db2-rs
-G IBM.ServiceIP:db2_db2inst2_0-rs-ip
```

```
(0) root@node1 # mkre1 -p DependsOn -S IBM.Application:db2_db2inst2_0-rs -G
IBM.Application:db2mnt_db2inst2_0_db2home-rs
(0) root@node1 # mkre1 -p DependsOn -S IBM.Application:db2_db2inst2_0-rs -G
IBM.Application:db2mnt_db2inst2_0_db2-rs
```

---

Figure 8-6 shows the relationships among the resources we create. These resources are related with the DependsOn relationship that has the following characteristics:

- ▶ Both source and target resources run on the same node.
- ▶ The source resource depends on the target resource.
- ▶ The source resource is started after the target resource are online.
- ▶ When the target resource fail, the source resource should also be stopped.

The start up sequence of the resource group db2\_db2omst2\_0-rs is as follows:

1. Select one network interface from the network equivalency resource for binding the service IP address.
2. Start the service IP address.
3. Start the LVM resource.
4. Start the mount point resources.
5. Start the DB2 resource.

This configuration assures that these resources are always run on the same node. If one of the resources fails, all the resources are moved to the failover node.

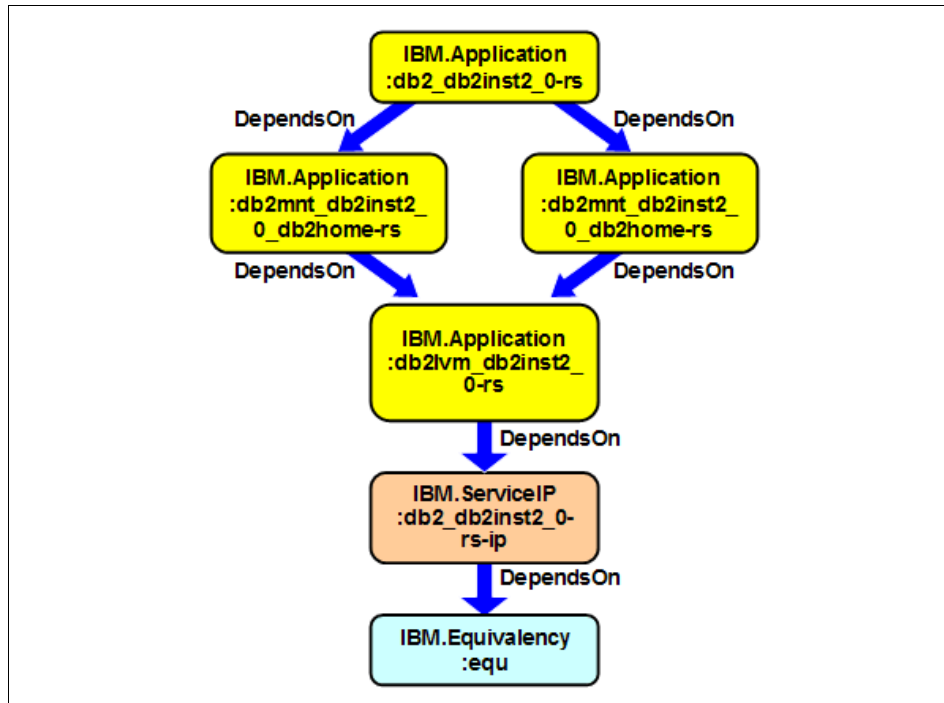


Figure 8-6 The relationships between the resources

### Check the cluster domain status

The `lssam` command shows the status of the resource groups. The `lssam` output shown in Example 8-31 provides information about the current state and the approximate state of the resource groups and the resources. We call the current state “OpState” and the approximate state “Nominal state”. The “Offline” items in the output are read as follows:

- ▶ The first state “Offline” on the farthest left (for example, “offline IBM.ResourceGroup:db2\_db2inst2\_0-rg”) means that the OpState of the resource group is offline.
- ▶ The second state “Offline” from the left (for example, “offline IBM.ServiceIP:db2\_db2inst2\_0-rs-ip”) means that the cluster wide OpState of each resource is offline.
- ▶ The third state “Offline” from the left (for example, “offline IBM.ServiceIP:db2\_db2inst2\_0-rs-ip:node1”) means that the node level OpState of each resource is offline.

The “Nominal” state on the right side of the first line indicates the to-be status of the resource group. The user specifies the expected status to TSA using the **chrg** command. TSA tries to maintain the status. For example, the nominal state of this resource group is offline now. If a resource is made to online manually (vary on the volume group and so on), TSA tries to stop the resource to make it offline. If the user issues the **chrg -o online <resource group>** command and change the nominal state to online, TSA will then try to start all the resources in this resource group.

*Example 8-31 The initial status of the resource group*

---

```
(0) root@node1 # lssam
Offline IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Offline
  |- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip
    |- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node1
    '- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node2
  |- Offline IBM.Application:db2_db2inst2_0-rs
    |- Offline IBM.Application:db2_db2inst2_0-rs:node1
    '- Offline IBM.Application:db2_db2inst2_0-rs:node2
  |- Offline IBM.Application:db21vm_db2inst2_0-rs
    |- Offline IBM.Application:db21vm_db2inst2_0-rs:node1
    '- Offline IBM.Application:db21vm_db2inst2_0-rs:node2
  |- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs
    |- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node1
    '- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node2
  '- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs
    |- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node1
    '- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node2
```

---

**Start the cluster domain**

Issue the **chrg** command to change the nominal state of a resource group to online:

```
(0) # chrg -o online <resource group>
```

TSA brings the resources up based on the defined the sequence. When you use the **lssam** command to view the resource status, you might see that some resources have been brought up and have the Online status. You also might see a *Pending Online* OpState indicating that the resource is on the way up.

Example 8-32 shows the output from the **chrg** and **lssam** commands. The nominal state is *Online*. TSA is trying to bring up the required resources and the *Pending Online* OpState showing that TSA’s effort is in progress. Most of the required resources are up and have the *Online* state except db2\_db2inst2\_0-rs. The startup process for this resource is still ongoing, therefore the state is *Pending Online*.

*Example 8-32 Sample chrg command and the Pending Online state*

---

```
(0) root@node1 # chrg -o online db2_db2inst2_0-rg
(0) root@node1 # lssam
Pending online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip
|  |- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node1
|  ' - Online IBM.ServiceIP:db2_db2inst2_0-rs-ip:node2
|- Pending online IBM.Application:db2_db2inst2_0-rs
|  |- Offline IBM.Application:db2_db2inst2_0-rs:node1
|  ' - Pending online IBM.Application:db2_db2inst2_0-rs:node2
|- Online IBM.Application:db2lvm_db2inst2_0-rs
|  |- Offline IBM.Application:db2lvm_db2inst2_0-rs:node1
|  ' - Online IBM.Application:db2lvm_db2inst2_0-rs:node2
|- Online IBM.Application:db2mnt_db2inst2_0_db2-rs
|  |- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node1
|  ' - Online IBM.Application:db2mnt_db2inst2_0_db2-rs:node2
' - Online IBM.Application:db2mnt_db2inst2_0_db2home-rs
|  |- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node1
|  ' - Online IBM.Application:db2mnt_db2inst2_0_db2home-rs:node2
```

---

Example 8-33 shows the `lssam` output when all the required resources in the resource group is up. The `OpState` of the resource group and the cluster wide resources are online. The node wide resources of node2 are online as well indicating that the current owner node of the cluster domain is node2.

*Example 8-33 The lssam output of the online resource group*

---

```
(0) root@node1 # lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip
|  |- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node1
|  ' - Online IBM.ServiceIP:db2_db2inst2_0-rs-ip:node2
|- Online IBM.Application:db2_db2inst2_0-rs
|  |- Offline IBM.Application:db2_db2inst2_0-rs:node1
|  ' - Online IBM.Application:db2_db2inst2_0-rs:node2
|- Online IBM.Application:db2lvm_db2inst2_0-rs
|  |- Offline IBM.Application:db2lvm_db2inst2_0-rs:node1
|  ' - Online IBM.Application:db2lvm_db2inst2_0-rs:node2
|- Online IBM.Application:db2mnt_db2inst2_0_db2-rs
|  |- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node1
|  ' - Online IBM.Application:db2mnt_db2inst2_0_db2-rs:node2
' - Online IBM.Application:db2mnt_db2inst2_0_db2home-rs
|  |- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node1
|  ' - Online IBM.Application:db2mnt_db2inst2_0_db2home-rs:node2
```

---



### **Perform a simple test: Move resource group**

We recommend doing this simple test to confirm that the cluster domain has been configured properly. Use the following command to move the resource group from the owner node to the failover node:

```
(0) # rgreq -o move <resource group name>
```

Example 8-34 shows the resource moving operation and the operational state in transition. Initially, node1 is online owning db2\_db2inst2\_0-rg resource group. We run **rgreq** to move the db2\_db2inst2\_0-rg resource group to node2. TSA should change node1 to offline and node2 to online. The cluster domain configuration is not complete if the db2\_db2inst2\_0-rg resource group failed to be moved.

#### *Example 8-34 Sample operation of resource group moving*

---

```
(0) root@node1 # lssam
```

```
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip
    |- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip:node1
    '- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node2
  |- Online IBM.Application:db2_db2inst2_0-rs
    |- Online IBM.Application:db2_db2inst2_0-rs:node1
    '- Offline IBM.Application:db2_db2inst2_0-rs:node2
  |- Online IBM.Application:db2lvm_db2inst2_0-rs
    |- Online IBM.Application:db2lvm_db2inst2_0-rs:node1
    '- Offline IBM.Application:db2lvm_db2inst2_0-rs:node2
  |- Online IBM.Application:db2mnt_db2inst2_0_db2-rs
    |- Online IBM.Application:db2mnt_db2inst2_0_db2-rs:node1
    '- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node2
  '- Online IBM.Application:db2mnt_db2inst2_0_db2home-rs
    |- Online IBM.Application:db2mnt_db2inst2_0_db2home-rs:node1
    '- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node2
```

```
(0) root@node1 # rgreq -o move db2_db2inst2_0-rg
```

```
Action on resource group "db2_db2inst2_0-rg" returned Token
"0xf82df5aac63c8a2d481a9302000b059f" .
```

```
(0) root@node1 # lssam
```

```
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip
    |- Offline IBM.ServiceIP:db2_db2inst2_0-rs-ip:node1
    '- Online IBM.ServiceIP:db2_db2inst2_0-rs-ip:node2
  |- Online IBM.Application:db2_db2inst2_0-rs
    |- Offline IBM.Application:db2_db2inst2_0-rs:node1
    '- Online IBM.Application:db2_db2inst2_0-rs:node2
  |- Online IBM.Application:db2lvm_db2inst2_0-rs
    |- Offline IBM.Application:db2lvm_db2inst2_0-rs:node1
    '- Online IBM.Application:db2lvm_db2inst2_0-rs:node2
  |- Online IBM.Application:db2mnt_db2inst2_0_db2-rs
    |- Offline IBM.Application:db2mnt_db2inst2_0_db2-rs:node1
    '- Online IBM.Application:db2mnt_db2inst2_0_db2-rs:node2
```

```
'- Online IBM.Application:db2mnt_db2inst2_0_db2home-rs
  |- Offline IBM.Application:db2mnt_db2inst2_0_db2home-rs:node1
  '- Online IBM.Application:db2mnt_db2inst2_0_db2home-rs:node2
```

---

## 8.5 Considerations for the db2nodes.cfg file

From DB2 Version 8 onwards, DB2 Enterprise Server Edition (ESE) single and partitioned database environments have a db2nodes.cfg file in the sqllib subdirectory of the Instance home. The db2nodes.cfg file is used to define the database partition servers that participate in a DB2 instance. In the cluster environment, TSA automatically takes care of the entry of this file to start DB2 instances on different nodes by updating the host name field.

Prior to Version 9.5, if a DB2 instance was set up in an HA configuration, the recommended method of starting the instance is to bring the corresponding resource group online through the cluster manager. This will no longer be required starting from DB2 9.5. However, DB2 can still be started this way for legacy reasons whether the CLUSTER\_MGR configuration parameter is set or not.

You can use the RESTART option of db2start to start the database manager after a failure. If neither the host name nor the logical-port parameter is specified, the database manager is restarted using the host name and logical-port values specified in db2nodes.cfg. If either parameter is specified, the new values are sent to the other database partitions when a connection is established. After db2start, the db2nodes.cfg file is updated with this information.

## 8.6 DB2 9.5 High Availability Feature

Starting from DB2 V9.5, TSA is bundled with the product and a utility called db2haicu is introduced for easier cluster configuration. The DB2 High Availability (HA) Feature provides infrastructure for enabling the database manager to communicate with the cluster manager when instance configuration changes, such as stopping a database manager instance or requiring cluster changes.

## 8.6.1 Cluster manager integration and interaction with DB2

The DB2 High Availability (HA) Feature is composed of these components:

- ▶ IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component:  
This is bundled with IBM Data Server on AIX and Linux and integrated with the DB2 installer. You can install, upgrade, or uninstall SA MP Base Component using either the DB2 installer or the installSAM and uninstallSAM scripts that are included in the IBM Data Server install media.
- ▶ DB2 High Availability Instance Configuration Utility (db2haicu):  
db2haicu is a text based utility that can be used to configure and administer highly available databases in a clustered environment. db2haicu collects information about the database instances, cluster environment, and cluster manager by querying the system. You supply more information through parameters to the db2haicu call, an input file, or at runtime by providing information at db2haicu prompts.
- ▶ DB2 cluster manager API:  
This API defines a set of functions that enable the database manager to communicate configuration changes to the cluster manager.

The DB2 High Availability (HA) Feature enables integration between IBM Data Server and cluster managing software from various perspectives. Figure 8-7 illustrates the integration of DB2 High Availability:

- ▶ Installation and Maintenance: IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component is bundled and installed as the default cluster manager on AIX and Linux. TSA FixPacks are installed as part of DB2 FixPacks.
- ▶ Configuration: The db2haicu utility allows you to set up and configure DB2 with cluster management.
- ▶ Operation: DB2 automatically updates the cluster manager when there are DB2 state changes significant to the cluster.
- ▶ Serviceability: The **db2pd -ha** option and messages in the db2diag.log allow for interaction with the cluster manager.

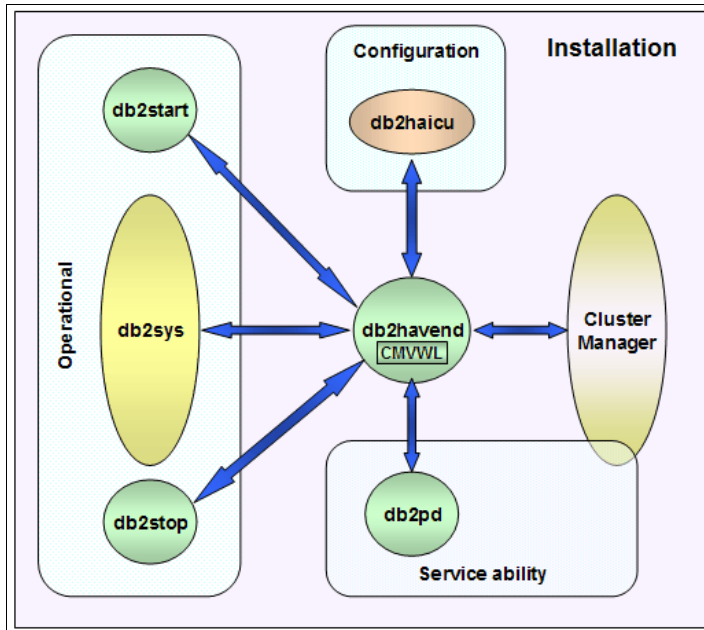


Figure 8-7 DB2 High Availability Integration

DB2 interacts with the cluster manager through a set of APIs. These APIs can be implemented by cluster manager vendors or any other third-party company and are represented by the cluster manager vendor wrapper library (CMVWL) shown in Figure 8-7. The DB2 High Availability Feature includes wrappers for TSA and HACMP. **db2havend** is a new process that is short lived for the duration required to complete the cluster manager commands.

## 8.6.2 Setting up TSA with DB2 9.5

IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component is bundled with IBM Data Server on AIX and Linux as part of the DB2 High Availability Feature, and integrated with the DB2 installer. It can be installed, upgraded, and uninstalled using either the DB2 installer or the `installSAM` and `uninstallSAM` scripts that are included in the IBM Data Server.

### TSA prerequisites

In DB2 9.5, the version of SA MP Base Component that is on the IBM Data Server install media is Version 2.2. It supports Linux on System z®, System x™, System i®, and System p®, as well as AIX 5.2 and AIX 5.3.

Other prerequisites exist in terms of IBM Reliable Scalable Cluster Technology (RSCT) versions or APARs on AIX, and libraries on Linux (for example, ksh must be installed and running). The TSA MP Base Component prerequisites are detailed in the Release Notes available at the following link:

[http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8216-02/en\\_US/PDF/HALRN208.pdf](http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8216-02/en_US/PDF/HALRN208.pdf)

During the installation of DB2, TSA prerequisites are checked by default.

You can also manually run the prereqSAM utility to check if TSA prerequisites are met. prereqSAM is located in the DB2 product package. For example, for ESE, it can be found under `~/ese/db2/linux/tsamp/prereqSAM`. Here is a successful run of prereqSAM:

```
mena:/software/V95/ese/db2/linuxamd64/tsamp # ./prereqSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on
operating system
SUSE Linux Enterprise Server 10 (x86_64)
VERSION = 10
PATCHLEVEL = 1
```

If the prerequisites are met, the product is installed. There is a prereqSAM.log file generated showing the successful run.

If the prerequisites are not met, a warning is logged in the installation log and a prereqSAM.<log-number>.log file is generated under the /tmp directory along with the db2\_install.log.<log-number> file. The log-number identifies the log file in the sequence. The DB2 installation will not fail.

## **Installing and upgrading the SA MP Base Component**

IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component can be installed or upgraded using either the DB2 installer or the installSAM install script that is included in the IBM Data Server.

### ***Using the DB2 installer***

The default behavior of the DB2 installer is to install or upgrade SA MP Base Component.

The following three methods are for using the DB2 installer:

► **DB2 Setup wizard:**

If the SA MP Base Component is not installed already or is at a lower level, during installation, the DB2 Setup wizard presents a panel titled “Install IBM Tivoli System Automation for Multiplatforms (SA MP) Base Component” for you to select the install option.

- ▶ Silent install using a response file with `db2setup`:  
To install or upgrade SA MP Base Component using a response file, set the response file keyword `INSTALL_TSAMP` to “YES”. To prevent the DB2 installer from installing SA MP Base Component in a response file installation, set `INSTALL_TSAMP` to “NO”.
- ▶ Command line utilities:  
Use `db2_install` command line utility to install, `installFixPack` command to upgrade, and `db2_deinstall` command to uninstall. To prevent `db2_install` from installing SA MP Base Component, use the `-f NOTSAMP` option with `db2_install`.

Regardless of which method you choose, the DB2 installer first queries your system for an already installed SA MP Base Component. If the SA MP Base Component is already installed, it will upgrade only when the current version is older than the version of SA MP Base Component that is bundled with DB2 V9.5 software, which is SA MP V2.2 FixPack 3.

You can use the `-l` option with `db2setup`, `db2_install`, or `installFixPack` to specify where the `installSAM` utility should place the SA MP Base Component install log.

### ***Using the installSAM install script***

The `installSAM` install script can be used to install or upgrade SA MP Base Component and is located on the IBM Data Server media at directory `db2/<platform>/tsamp`, where *platform* refers to the appropriate hardware platform. For information about using `installSAM` see:

<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>.

You cannot upgrade SA MP Base Component using either the DB2 installer or the `installSAM` install script if you have:

- ▶ SA MP Base Component Version 1 currently installed. You must upgrade from Version 1 to Version 2.1 before you can upgrade to Version 2.2.
- ▶ One or more IBM RSCT peer domains defined on your system.

After SA MP is installed or upgraded using the `db2` installer, HA policy scripts are located at `/usr/sbin/rsct/sapolicies/db2`. If you used the `installSAM` install script, then these HA policy scripts should be installed manually using the `db2cpts` utility.



## DB2 and HACMP

In this chapter we explain how to integrate DB2 in a High Availability Cluster Multi-Processing (HACMP) environment. The DB2 in the cluster is in a shared disk. We provide the basic management concepts with recommendations and considerations to reduce the time consumed for failover.

We cover the following topics:

- ▶ Overview
- ▶ How DB2 works with HACMP
- ▶ Planning an HACMP cluster
- ▶ Setting up an HACMP cluster
- ▶ Considerations for the db2nodes.cfg file
- ▶ Tuning tips for quick failover

## 9.1 Overview

HACMP for AIX provides a highly available computing environment. HACMP facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. The primary reason to create HACMP clusters is to provide a highly available environment for mission-critical applications. In an HACMP cluster, to ensure the availability of these applications, the applications are placed under HACMP control. HACMP ensures that the applications remain available to client processes even if a component in a cluster fails. To ensure availability in case of a component failure, the HACMP software moves the application along with resources to another node in the cluster.

Here we list some common HACMP terms:

- ▶ **Topology:**  
The layout of physical components and connections defined in HACMP.
- ▶ **Cluster:**  
The group of nodes that work together closely for the purpose of enhancing availability of services.
- ▶ **Node:**  
Each server within the cluster definition. A *Service (or Primary)* node is designated as active to provide service to applications. A *Standby* node sits ready to take over if the service node fails.
- ▶ **Resource:**  
A *resource* is an object that is protected and controlled by HACMP. It may include the IP address to which clients access, file systems, or raw devices on shared volume groups, and the start and stop scripts to control applications.
- ▶ **Resource group:**  
This is a group of all the resources that must be failed over from one server to the other. These include, but are not limited to, the following items:
  - The shared disks as defined in the volume groups. Raw devices or file systems are defined on them.
  - The IP address (*Service Address*) that the clients connect to.
  - The applications to be started to provide services. DB2 instance start and stop scripts are included in this definition.
- ▶ **Service address:**  
The IP address that provides services to clients. The service IP address is included in the resource group.



▶ Shared disk:

Shared disk is storage devices and volume groups connected to multiple nodes. Raw devices and file systems required to provide services are placed on them. From the perspective of DB2 in a non HADR implementation of HACMP, this would include the database directory, table space containers, database logs, and so on.

▶ Application server:

The application server is simply a set of scripts used to start and stop the applications running on HACMP node. One is an *application server start script* that starts up the applications which are a prerequisite to a provision of client services. The other is an *application server stop script* that stops applications before releasing resource groups. From the perspective of DB2, this would include scripts to start up and stop the database instance.

While there is only one main script to provide stop and start functionality for all applications and services, this work can easily be made modular by splitting the tasks into many subscripts which are called from the main script. For instance, with appropriate logic in the main calling script, subscripts can be called in a specific order according to file name, and can be dynamically added or removed from a subdirectory as required without changing the main calling script.

For more information about HACMP terminology, refer to *AIX HACMP v5.4 Master Glossary*, SC23-4867-08.

## 9.2 How DB2 works with HACMP

In this section, we focus on the HACMP cluster with databases created on shared disks. High availability is enabled by moving these disk resources (changing which node has control over them) and restarting database instance on a standby node when the service node has an outage. We call this classic type of cluster a *shared disk cluster*. Here we explain how DB2 works in a shared disk cluster controlled by HACMP with an example of a simple single partitioned instance.

Figure 9-1 illustrates HACMP topology and related DB2 database components. HACMP manages the group of resources required to provide a service, including shared disks where database components reside, the IP address for client service, and the application server handling the starting and stopping of the applications (including DB2 database instance).

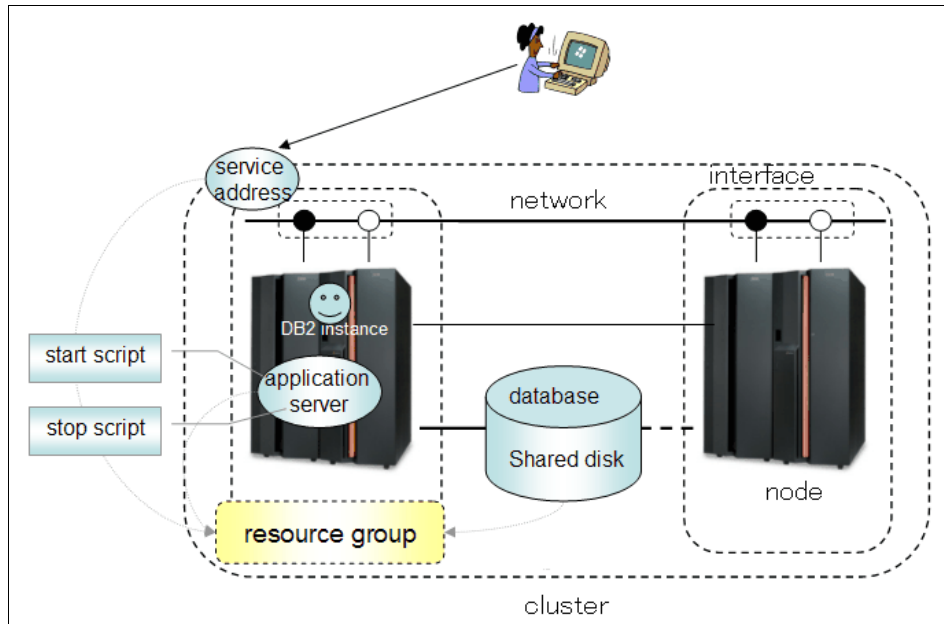


Figure 9-1 HACMP topology and related DB2 database components

HACMP continuously sends heartbeats between the nodes in the cluster to identify if and when the other one is down. HACMP handles resource group takeover from one system to the other as necessary after a failure or fallback after the failed node is repaired. In a DB2 single-partitioned environment, the cluster takes the following actions:

► *Failover* (unplanned takeover):

If the service node fails, the HACMP standby node detects the service node outage. When the *keepalive* packets from all network routes are not received from the other node, the standby node starts to take over the resource group. This means that the standby node acquires the shared storage devices and the volume group where database components are created, and the service IP address through which clients communicate with the server. After the resources are taken over, the start script defined in the application server is issued, which starts the DB2 instance along with other critical services and applications.

Figure 9-2 shows how the resource group has been moved to a standby node during failover.

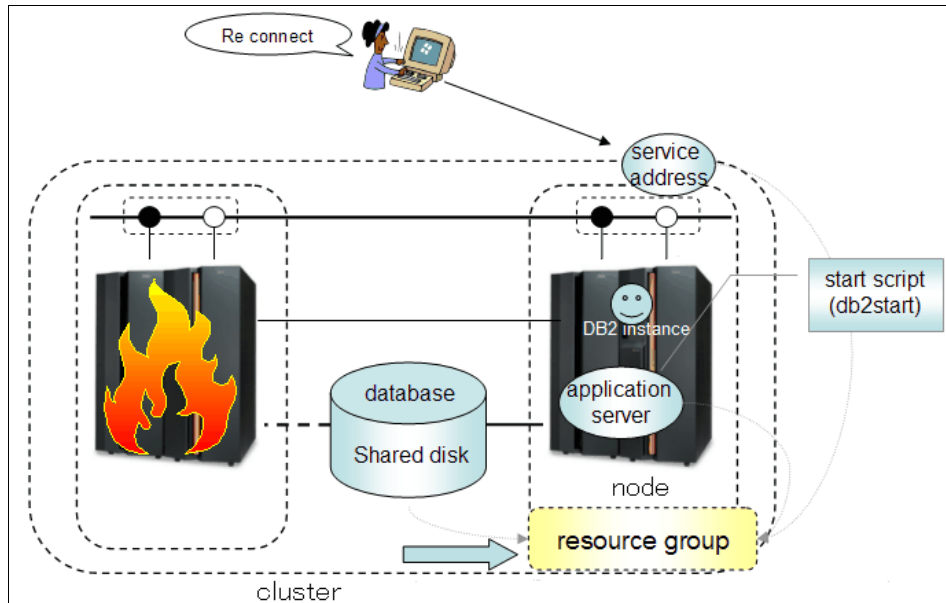


Figure 9-2 Failover (unplanned takeover)

► *Switch over* (planned takeover for maintenance):

You can switch over the resource group intentionally to the standby node for the purpose of machine maintenance. This is done by performing two HACMP management operations:

- Stop HACMP in takeover mode on service node.

The application server Stop script on the node that needs to release the resource group is invoked. The Stop script gracefully stops all the applications and processes which access the shared disk.

- Move the resource group from the service node to the standby node.

The HACMP process is still running on both nodes. HACMP releases all resources including shared disks and any service IP address related to the resource group. After released by the service node, the resource group is acquired on the standby node's side.

Figure 9-3 illustrates how the resource group moves during switch over.

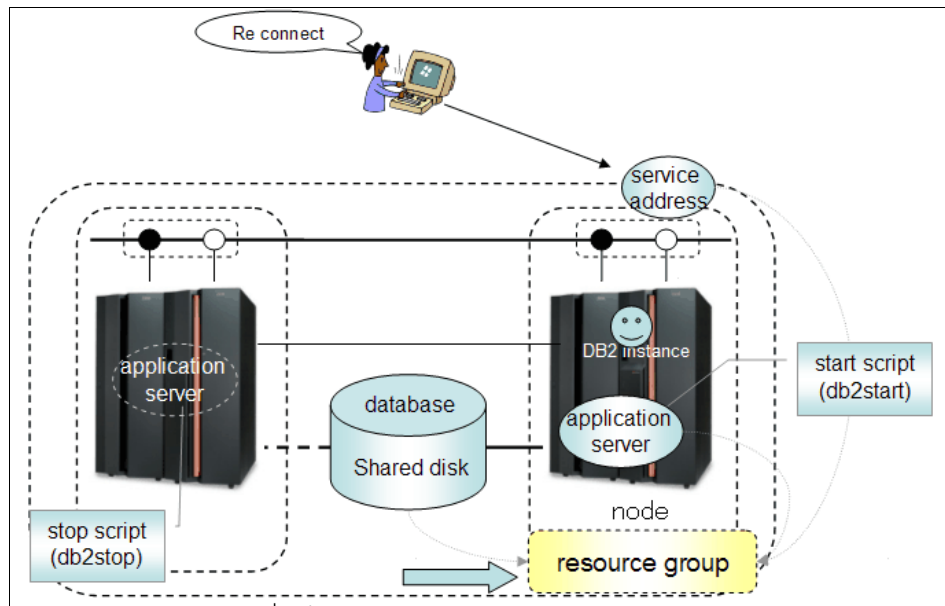


Figure 9-3 Switch over (planned takeover for maintenance)

## 9.3 Planning the HACMP cluster

Planning a HACMP cluster is similar to planning a high availability cluster with Tivoli System Automation (TSA). Refer to 8.3, “Planning the high availability cluster” on page 241.

## 9.4 Setting up HACMP

Setting up HACMP usually requires the following steps:

1. Define the cluster and add nodes to the cluster.
2. Configure the HACMP network and decide which method is to be used for IP address takeover.
3. Configure the communication interface and devices. A heart-beating disk can be used as the heart beat device. To configure the heart-beating disk, the shared disk should be in an enhanced concurrent volume group.
4. Configure the high availability service IP.

5. Configure the application server.

It is often useful to place a simple test script (dummy script) for the corresponding start and stop script entry to test the basic functions of HACMP. After verifying that the HACMP environment functions properly, modify the start and stop script to work.

6. Configure the High Availability (HA) resource group.

This will associate the HA service IP, shared volume groups and files systems, the application server, and all the resources required by the application into one resource group. In case a node failure happens, the HACMP will move the resource group running on the failed node to the surviving node in the cluster.

7. Verify and synchronize HACMP configuration.

After configuring HACMP, you must verify and synchronize the topology and resource group using the facility provided by HACMP (for example, via SMIT command panels). Every time the HACMP configuration is changed, you must re-synchronize the cluster from the node where the change has been made.

After the verification and synchronization is successful, you can start the HACMP service.

## 9.4.1 HACMP cluster setup planning

Before you set up an HACMP environment, you need to plan the cluster environment. Consider the following list of items:

- ▶ Physical nodes
- ▶ Network
- ▶ Application back end/services software (for example, DB2)
- ▶ HACMP configuration
- ▶ Application server start/stop scripts

Next, we examine the planning of these items briefly in a sample environment specific to a disk sharing configuration. Detailed information and instructions for each item can be found in the links provided in 9.4.2, “HACMP configuration” on page 285.

## Sample environment

Figure 9-4 shows the sample configuration of an active/standby HACMP cluster configuration, with a normal DB2 instance in a shared disk configuration.

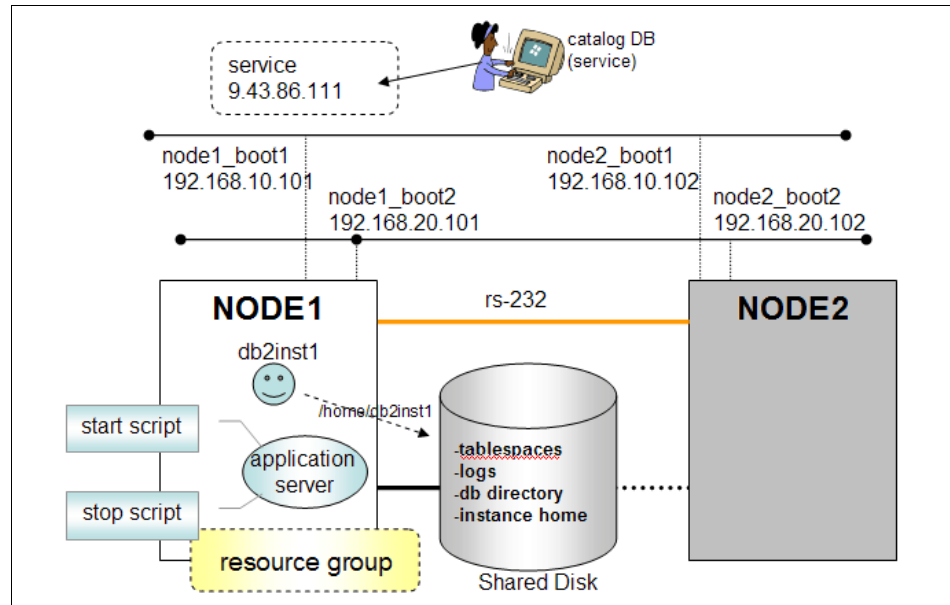


Figure 9-4 HACMP disk shared cluster

Here is the planning we did for this sample environment:

- ▶ Physical node configuration:
  - Two physical nodes named Node1 and Node2 are defined in the cluster:
    - Node1 is designated as the *service node* normally providing service for clients. A single partitioned DB2 instance is running on this node.
    - Node2 is the *standby node*, which normally stands by for any service node outage. No DB2 instance runs on the standby node. The role of both nodes changes in response to system failover, or to planned takeover issued by administrators.
- ▶ Network configuration:
  - Two ethernet network interfaces on each node are provided for client connectivity. These are both under the control of HACMP. The service address for clients is added on one of these network interfaces.
  - One Serial (RS-232C) network is configured for HACMP keepalive. A serial network (non-TCP/IP network) is recommended, as it makes HACMP failure detection more reliable.

► DB2 configuration:

The DB2 product libraries are installed on the local disk of each server. A single partitioned instance named *db2inst1* and its database named SAMPLE are created on a shared disk. If table spaces or logs are placed in different devices within the shared disk, those devices would all have to be included in a single resource group.

► HACMP configuration:

- A resource group named *shared\_rg* is configured in the cluster, which has a service IP address, application server, and file systems on a shared disk as resources.
- A service address is defined in the resource group. Each DB2 client has the entry for this service address in its catalog node directory and connects to the Service node through this IP address.
- An application server is also defined in resource group. The *application server start script* simply starts up the database instance and restarts the database. The *application server stop script* stops the instance.

## 9.4.2 HACMP configuration

In the last section, we briefly discussed the planning stage prior to HACMP implementation. In this section we introduce an outline of actually setting up HACMP in a shared disk cluster configuration.

### Checking the network connection

Before configuring HACMP, the network must be configured properly. The following steps show how to check network configurations for the cluster.

1. Check that IP addresses are configured on network interfaces on both nodes.
2. Check that the */etc/hosts* file has all entries of IP addresses and their labels are those used by HACMP.
3. Verify that the name resolution is working well using the **host** command. If something is wrong, check and modify the */etc/hosts* file.
4. Check the serial network connection. The subsidiary network for HACMP keepalive is recommended to make HACMP failure detection more secure. For more detail, see *HACMP Administration Guide*, SC23-4862-09.

### Configuring the shared disk

This requirement is specific to an HACMP shared disk configuration. Because shared disks are such an integral part of the HACMP setup, this section lists the high level steps needed to set up shared disk resources.

Following are some of the terms concerning AIX Logical Volume Manager, which are used in this section:

- ▶ Volume group (VG)
- ▶ Logical volume (LV)
- ▶ Journalled file system (JFS or JFS2)
- ▶ File system (FS)
- ▶ Log that maintains a consistent JFS (JFSLog or JFS2LOG)

Perform these steps to set up shared disk drives and the logical volume manager:

1. Check the disk drives.

Check that the external disk is configured and recognized from both nodes.

2. Create the VG.

Create the VG on the service node. The VG must have a unique name and major number (serial ID of VG) for all nodes in the cluster. You can check available major numbers on both nodes with the `lvfstmajor` command:

```
root@node1:/# lvfstmajor
58...
```

To add a volume group, you can use the `smitt` menu:

```
#smitty vg
```

From **Volume Groups** → **Add a Volume Group** → **Add an Original Volumes Group**. Enter *VOLUME GROUP name*, and the available *Volume group MAJOR NUMBER*. as shown in Example 9-1.

*Example 9-1 Add volume group*

---

Add an Original Volume Group

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

```

VOLUME GROUP name                                [Entry Fields]
Physical partition SIZE in megabytes             [db2vg]
+
* PHYSICAL VOLUME names                           [hdisk1]
+
Force the creation of a volume group?            no
+
Activate volume group AUTOMATICALLY              yes
+
  at system restart?
```



```
Volume Group MAJOR NUMBER          []
+#
Create VG Concurrent Capable?      no
+
```

---

Activate the newly created VG by issuing the following command:

```
# varyonvg db2vg
```

### 3. Create the JFS or JFS2 log.

Create the JFS or JFS2 log with a unique name on the new VG. When creating the first file system on a new VG, AIX will automatically create a JFS or JFS2 log, with the name of `loglv00`, `loglv01`, and so on, for each new JFS or JFS2 log on the machine. By default, AIX creates only one JFS and JFS2 log per VG. Because a unique name is needed for the JFS or JFS2 log, it is best to define the JFS or JFS2 log with the `mk1v` command, before creating the first file system.

To create a JFS2 log with the name `db2vgjfslog` in the VG `db2vg`, issue the following command:

```
# mk1v -t jfs2log -y db2vgjfslog db2vg 1
```

To format the JFS2 Log, issue the following command, and select `y` when asked whether to destroy the LV:

```
# logform /dev/db2vgjfslog
```

```
logform: destroy /dev/db2vgjfslog (y)? y
```

**Note:** HACMP 5.x supports JFS2 with inline logs. Only note you should install the fix for the following APAR:

- ▶ HACMP V5.2: IZ05931
- ▶ HACMP V5.3: IZ16032
- ▶ HACMP V5.4: IZ08317

### 4. Create a file system.

Create any LVs and JFSs that are needed, and ensure that they have unique names and are not currently defined on any node. Set the FSs so that they are not mounted on restart. Verify the current LV and JFS names.

To create the JFS2 LV for the `/home/db2inst1` file system, issue the following command:

```
# smit 1v
```

In the Add a Logical Volume menu, select a VG name and press **Enter**. The Add a Logical Volume menu is presented. Fill in the fields as shown in Example 9-2.

Example 9-2 Add logical volume

---

Add a Logical Volume

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

```
[TOP]                                     [Entry Fields]
  Logical volume NAME                       [home1v01]
* VOLUME GROUP name                         db2vg
* Number of LOGICAL PARTITIONS             [16]
#
  PHYSICAL VOLUME names                    []
+
  Logical volume TYPE                       [jfs2]
  POSITION on physical volume                middle
+
  RANGE of physical volumes                 minimum
+
  MAXIMUM NUMBER of PHYSICAL VOLUMES       []
#
  to use for allocation
  Number of COPIES of each logical         1
+
  partition
  Mirror Write Consistency?                active
+
  Allocate each logical partition copy      yes
+
  on a SEPARATE physical volume?
  RELOCATE the logical volume during        yes
+
  reorganization?
[MORE...9]
```

---

After the LV has been created, we can create a file system associated with this LV. In this example, we create a JFS2 file system. To add a JFS2 file system on the previously defined LV, issue the following command:

```
# smitty jfs2
```

Select **Add an Enhanced Journaled File System on a Previously Defined Logical Volume** and fill in the fields. See Example 9-3.

---

Add an Enhanced Journalled File System

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

```

                                                    [Entry Fields]
* LOGICAL VOLUME name                               home1v01
+
* MOUNT POINT                                       [/home/db2inst1]
Mount AUTOMATICALLY at system restart?           no
+
PERMISSIONS                                         read/write
+
Mount OPTIONS                                       []
+
Block Size (bytes)                                 4096
+
Logical Volume for Log
+
Inline Log size (MBytes)                           []
#
Extended Attribute Format                           Version 1
+
ENABLE Quota Management?                           no
+
```

---

After a file system is created, it is not automatically mounted. Check that the file system can be mounted manually with the following command:

```
#mount /home/db2inst1
```

5. Unmount all of the FSs and deactivate the VG.

To do this, invoke the following commands:

```
#umount /home/db2inst1
#varyoffvg db2vg
```

6. Import the VG.

Import the VG on the standby node (node2) with the same major number, and change the VG so that it is not activated on restart. When the VG is imported on the node2, the definitions of the FSs and LVs are imported to node2. If we ever need to NFS export the file system, the major number for the VG has to be identical on both nodes. Make sure that the VG is defined not to be activated automatically on reboot, because it can be activated and controlled by HACMP.

To import a VG, issue the following command:

```
# smit vg
```

Select **Import a Volume Group** and fill in the fields. See Example 9-4.

*Example 9-4 Importing a volume group*

---

Import a Volume Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

```

                                     [Entry Fields]
VOLUME GROUP name                    [db2vg]
* PHYSICAL VOLUME name                [hdisk1]
+
Volume group MAJOR NUMBER             [58]
+#
```

---

Next, change the VG so that it is not activated on reboot:

```
# smit vg
```

Select **Set Characteristics of a Volume Group** → **Change a Volume Group**. Then select **VOLUME GROUP name**. In the Change a Volume Group menu, select **no** on Activate volume group AUTOMATICALLY.

7. Move the VG back to the service node.

Move the file system and VG back to node1 for the next step where we will create a DB2 instance user on node1.

```
# umount /home/db2inst1
# varyoffvg db2vg
```

Next, run the following commands on node1:

```
# varyonvg db2vg
# mount /home/db2inst1
```

## User/group setup and DB2 installation

Now that the shared disk is set up, we can create the DB2 instance and database on the shared disk. On node1, perform the following the steps:

1. Create the group for the DB2 instance.  
Use the **mkgroup** command to create the group.
2. Create the user for the DB2 instance.  
Use the **mkuser** and **passwd** commands.

3. Change the ownership of the FSs and LVs.
4. Install DB2 and set up the license key. The DB2 product has to be installed on the local disk of both nodes.
5. Create the DB2 instance.
6. Create a database in the shared file system.
7. Fail instance home directory over to node2.

Unmount all the FSs on the shared disk and varyoff the VG on node1.  
Activate the VG and mount all the FSs on node2.

Next, on node2, delete the directory `/home/db2inst1/sqllib` to allow us to create the instance and control files on node 2, then repeat steps 1 to 5. Using this procedure allows DB2 to automatically perform steps such as adding service entries and modify `db2nodes.cfg` file for you. After that, recatalog the database using the following command:

```
db2 catalog database dbname on <DBPATH>
```

Instead of deleting `/home/db2inst1/sqllib`, you can simply repeat steps 1 to 4 on node2, then manually create the Instance user IDs and group IDs on the second node, specifying the existing home directory in the shared file system. Make sure that `/etc/services` contains the correct DB2 port assignments and update the `db2nodes.cfg` with an alias known to both nodes. This alias must be added to the `/etc/hosts` file on each node.

## Preparing application server scripts

The application server is composed of a start script and a stop script. In this section we explain what has to be included in these application server scripts.

- ▶ In the start scripts, we simply restart the DB2 instance and activate the databases to run crash recovery.
  - DB2 Enterprise Server Edition (ESE) instance has the `db2nodes.cfg` file in the `sqllib` subdirectory of the instance home. This file must contain the correct host name and IP address when you restart the DB2 instance on the other node. DB2 single partition instances also have this file. We explain this topic in more detail in 9.5, “Considerations for `db2nodes.cfg` file” on page 295.
  - Set the database configuration parameter `AUTORESTART` to `ON`. The database manager automatically calls the restart database utility, if required, when an application connects to a database or a database is activated. The default setting is `ON`.

- ▶ In the stop scripts, we have to stop all applications and processes which access the shared disk to ensure that the node can release the shared disk successfully. In DB2 terms, this means **force application all**, **deactivate** for all databases, **terminate** to stop the back end process, and **db2stop** to stop the dbm/instance. Escalation commands such as **db2stop force** and **db2\_kill** may be necessary in order to get applications disconnected in a reasonable period of time.

A sample script file is packaged with the DB2 ESE for AIX to assist in configuring for HACMP failover or recovery in either hot standby or mutual takeover nodes. The script file is called *rc.db2pe.ee* for a single node (other than ESE) and *rc.db2pe.eee* for multiple nodes. They are located in the `sqlib/samples/hacmp/es` subdirectory of the DB2 Instance home. The appropriate file can be copied and customized for your system. When customized and renamed, rename `rc.db2pe.ee` to `rc.db2pe`. For example, these sample scripts are designed to be called with syntax `rc.db2pe db2inst1 start`, and `rc.db2pe db2inst1 stop`, respectively.

**Note:** *rc.db2pe.eee* is based on Parallel System Support Programs (PSSP) for an AIX environment; you cannot use it directly if there is no PSSP environment. However, you can customize it to adapt your requirements.

For more information about HACMP/ES events, refer to the section on HACMP ES Event Monitoring and User-defined Events in the Cluster Support for AIX chapter of the DB2 9 manual *Data Recovery and High Availability Guide and Reference*, SC10-4228. This topic is also available in the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0007500.htm>

## Configuring HACMP

AIX `smitty` provides HACMP configuration interfaces. The main steps are:

1. Add nodes to the HACMP cluster
2. Add a service IP label/address
3. Configure application servers
4. Add a resource group
5. Add resources to the resource group
6. Define serial network and serial network device
7. Verify and synchronize HACMP configurations
8. Basic verification of HACMP configuration

Here we expand upon each step:

1. Add nodes to HACMP cluster.

```
#smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Add Nodes to an HACMP Cluster**

In Configure Nodes to an HACMP Cluster (standard) menu, enter the cluster name and new nodes.

2. Add a service IP address.

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Service IP Labels/Addresses** → **Add a Service IP Label/Address**

In the Add a Service IP Label/Address menu, enter the IP label and network name

3. Configure application servers.

To access this menu, proceed as follows:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Application Servers** → **Add an Application Server**

In the Add application Server menu, enter the server name, and complete paths of the start and stop scripts.

**Note:** The start and stop scripts that are called from the application server have to exist in the local directory on both nodes and have the same name.

4. Add a resource group.

To add a resource group, go to **smitty**:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Add a Resource Group**

In the Add a Resource Group menu, enter the resource group name and participating node names.

5. Add resources to the resource group:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Change/Show Resources for a Resource Group (standard)**

In the Change/Show Resources for a Resource Group menu (Example 9-5), enter the service IP label/address, application server name, volume groups, and file system information.

The resource group policies we set in this example are as follows. This corresponds to the *Rotating resource group* in former HACMP versions, which means there is no priority for resource groups between nodes. For further details, see Chapter 6 in *HACMP v5.3 Planning and Installation Guide*, SC23-4861-07.

*Example 9-5 Adding a resource to a resource group*

---

Change/Show All Resources and Attributes for a Resource Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]
Resource Group Name	db2_rg
Participating Nodes (Default Node Priority)	node1_bt node2_bt
Startup Policy	Online Using
Distribution Policy	
Fallover Policy	Fallover To Next
Priority Node In The List	
Fallback Policy	Never Fallback>
Service IP Labels/Addresses	[service]
+ Application Servers	[db2_server]
+ Volume Groups	[db2vg]
+ Use forced varyon of volume groups, if necessary	false
+ Filesystems (empty is ALL for VGs specified)	[/home/db2inst1 ]
+	

---



6. Define the serial network and serial network device.

If you have a serial network as the subsidiary keepalive network, you can configure this from the **smit** menu.

7. Verify and synchronize HACMP configurations.

After defining the HACMP configuration from one node, you must verify and synchronize the cluster topology to the other node.

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Verify and Synchronize HACMP Configuration**.

HACMP verification utility checks that the cluster definitions are the same on all nodes and provides diagnostic messages if errors are found.

The HACMP configuration details can be found in the following publications:

- ▶ *High Availability Cluster Multi-Processing for AIX* PDF manual links:  
[http://www-03.ibm.com/systems/p/library/hacmp\\_docs.html](http://www-03.ibm.com/systems/p/library/hacmp_docs.html)
- ▶ *HACMP v5.4 Administration Guide*, SC23-4862-09
- ▶ *HACMP v5.3 Administration Guide*, SC23-4862-06
- ▶ *HACMP v5.3 Planning and Installation Guide*, SC23-4861-07
- ▶ *Data Recovery and High Availability Guide and Reference*, SC10-4228

## 9.5 Considerations for db2nodes.cfg file

From version 8 onwards, DB2 ESE single and partitioned instances have a `db2nodes.cfg` file in the `sqllib` subdirectory of the Instance home. The `db2nodes.cfg` file is used to define the database partition servers that participate in a DB2 instance. In the cluster environment, you have to consider the entry of this file to start DB2 instances on different nodes.

Suppose that a DB2 ESE single partition instance is running in a cluster that is composed of a service node named `node1` and a standby node named `node2`. When a takeover occurs, the DB2 instance has to be started on `node2`. If the Instance home directory is configured on the shared disk, the entry in `db2nodes.cfg` file on the shared directory will not match the host name of the standby node. Then the **db2start** command fails with error codes like -6048 or -6031 when the application start script tries to start the DB2 instance. Example 9-6 shows the case of **db2start** failing when `db2nodes.cfg` does not match the host name.

*Example 9-6 Error messages caused by an invalid db2nodes.cfg entry*

---

```
$hostname
host2

$cat /home/db2inst1/sqllib/db2nodes.cfg
0 host1 0

$ db2start
SQL6048N A communication error occurred during START or STOP DATABASE
MANAGER processing.
```

---

You have several options to avoid this error, some of which we list here:

1. Modify the file entry in the start script.
2. Use the **db2start** command with the **restart** option.
3. Use the **db2gcf** command with the **-u** option.
4. Use an alias in the hosts files.

Options 1, 2, and 3 modify the entry of the db2nodes.cfg file manually or automatically, while options 4 does not. Options 2 requires permission for remote execution, while options 1, 3, and 4 do not.

Next we describe each of these options in further detail. The following DB2 Information web pages also contains information about the format of the DB2 node configuraiton file:

- ▶ DB2 9.5:  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.1uw.qb.server.doc/doc/r0006351.html>
- ▶ DB2 9.7:  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.1uw.qb.server.doc/doc/r0006351.html>

### **Modifying the file entry in the start script**

Perhaps the simplest of the listed methods, this entails modifying the entry of the db2nodes.cfg file before starting the DB2 instance, or preparing another configuration file with the correct entry and overwriting the existing db2nodes.cfg file. In our example, a DB2 instance, on the HACMP Service node named node1, (node2 as standby), has a db2nodes.cfg file shown here:

```
0 node1 0
```

In the event of a takeover on node2, the db2nodes.cfg file shown previously would be invalid. Therefore, the application start script can include a process to modify db2nodes.cfg as shown next, or prepare a local file that already has the

following entry and that copies or overwrites this file to the db2nodes.cfg file before actually starting the DB2 instance on node2:

```
0 node2 0
```

Figure 9-5 shows this concept graphically, with states shown before, during, and after the takeover by node 2 has occurred.

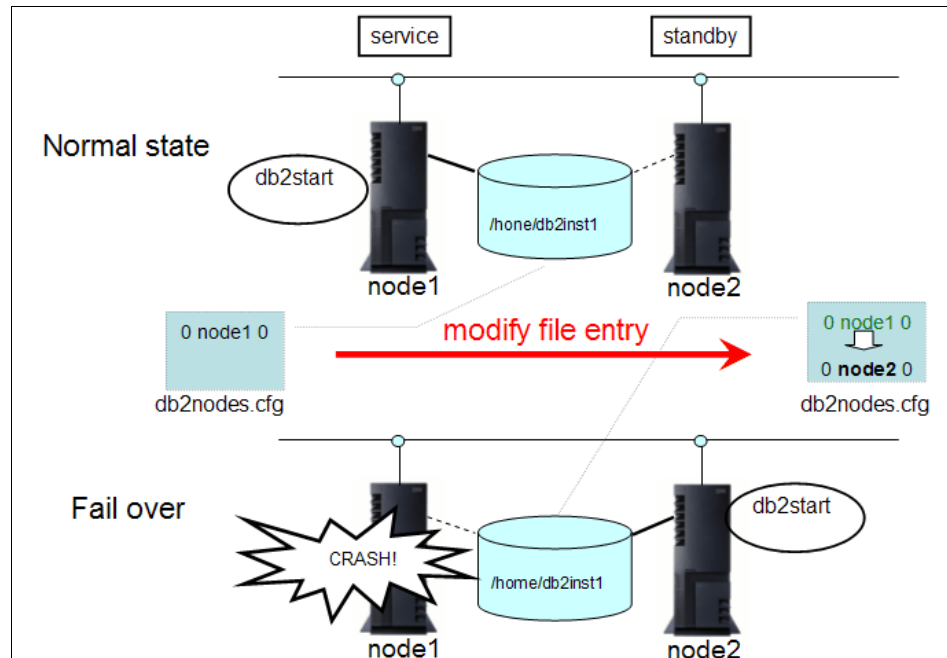


Figure 9-5 Modifying the db2nodes.cfg entry before starting DB2 on node2

### Using the db2start command with the restart option

Starting the DB2 instance with the restart option gives you the ability to specify the new host name as the node that is taking over the resource groups. See Example 9-7. Note that DB2 automatically modifies the db2nodes.cfg file with the host name you specified in the restart option.

*Example 9-7 Restart option of the db2start command*

```
db2start dbpartitionnum 0 restart hostname node2
```

### Considerations

Because the restart option requires the permission of remote execution, you have to configure remote shell (rsh) or secure shell (ssh) to be available in the cluster. This is required for both single and multiple node partitions.

- ▶ rsh configuration:  
Add entries of a reliable host name and user to the .rhosts file or hosts.equiv file. This option is often not viewed favorably due to the security risks that the remote shell might present.
- ▶ ssh configuration (available from DB2 V8.2):  
Install ssh, set the public key and private key for the use of the DB2 instance. Set full path of `ssh` command in DB2RSHCMD registry value as follows:  
`db2set DB2RSHCMD=/usr/bin/ssh`

### Using the `db2gcf` command with the `-u` option

This option uses the `db2gcf` command. When issuing the `db2gcf` command with the `-u` option, the `db2nodes.cfg` file is automatically modified and the DB2 instance starts on the standby node. With this method, rsh or ssh are not required, so the method is useful if you have a site or company policy against using any .rhosts files for security reasons, or if you cannot have ssh installed.

See Example 9-8 for a `db2gcf` command example and Example 9-9 for the `db2nodes.cfg` file modified by this command. You might notice in Example 9-9 that a fourth parameter has been appended to `db2nodes.cfg`. This is *Netname*, used for interconnect for FCM communication. For more details, refer to *Quick Beginnings for DB2 Servers*, GC10-4246.

*Example 9-8 db2gcf command issued with the -u option*

---

```
$ db2gcf -u -p 0 -i db2inst1
```

```
Instance : db2inst1
DB2 Start : Success
Partition 0 : Success
```

---

*Example 9-9 Entry of db2nodes.cfg file modified by the db2gcf command*

---

```
0 node2 0 node2
```

---

For more details on `db2gcf`, refer to *Command Reference*, SC10-4226, or the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/r0010986.htm>

### Using an alias in the hosts file

This option uses an alias in the `/etc/hosts` file. The entry of the `db2nodes.cfg` file never changes during takeover. With this method, rsh or ssh are *not required*, so

the method is useful if you have a site or company policy against using any .rhosts files for security reasons, or if you cannot have ssh installed.

You have to create an alias entry in the /etc/hosts file on both systems with the following format:

```
ip_address short_name long_name alias
```

On node1 (primary system), the entry looks similar to Example 9-10.

*Example 9-10 /etc/hosts file of node1*

---

```
192.168.10.101    node1    node1.itsosj.sanjose.ibm.com    db2host
```

---

On node2 (standby system), the entry looks similar to Example 9-11.

*Example 9-11 /etc/hosts file on node2*

---

```
192.168.10.102    node2    node2.itsosj.sanjose.ibm.com    db2host
```

---

The IP address and domain name for both files should match network definitions for the primary and standby systems, that is, the alias goes against the actual local host name entry, as DB2 uses the host name as the basis for what should be in db2nodes.cfg before it will start.

Next, the hosts entry in the /etc/netshvc.conf file must contain *local* as the first parameter. The hosts entry on both primary and standby systems would then look similar to Example 9-12.

*Example 9-12 /Entry of etc/netshvc.conf file*

---

```
hosts=local,bind
```

---

By putting local as the first parameter, this will force the system to look in the /etc/hosts file for a host name entry before going to the Domain Name Server (DNS).

Finally, parameter/column two in the db2nodes.cfg file is changed to the alias defined on both systems. The new db2nodes.cfg file then resembles Example 9-13, where db2host is the name defined in the hosts file on both systems.

*Example 9-13 Entry of db2nodes.cfg file*

---

```
0 db2host 0
```

---

After completing these steps, failover and fallback testing should be done to ensure the HA computing environment is working correctly.

## 9.6 Tuning tips for quick failover

When an outage on a primary database occurs, the steps required for continuing database services are as follows:

1. Failure detection
2. Resource failover required to provide service
3. Restart applications and services

To speed up the recovery time, we need to reduce the time taken in each step. In this section we discuss the considerations for each step. See Figure 9-6.

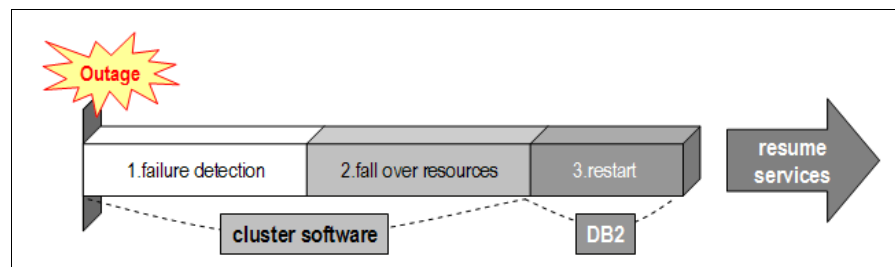


Figure 9-6 Time consumed by failover process

### 9.6.1 Failure detection time

Failure detection time can be tuned by the cluster software settings. For HACMP, *failure detection rate* for each network type can be specified. There are several network types: atm, ethernet, fddi, hps, token, diskhb, rs232, tmssci, and tmssa. We recommend the same detection rate through all networks in the cluster.

HACMP provides three default settings: SLOW(48s), NORMAL(20s), and FAST(10s). You can also specify a custom failure detection rate in seconds. Note that setting a very low number of seconds can cause misjudgment of failure detection (false positives), resulting in unnecessary frequent failover.

### 9.6.2 Failover of the resources

In this section, we describe several options to speed up resource failover time.

## IP address takeover

In previous versions of HACMP, network interfaces were directly overwritten by the service address. From HACMP/ES V4.5, an IP alias takeover is available, which reduces time for adding the service IP address to the standby node. As stated in the *HACMP v5.3 Administration Guide*, SC23-4862-06, the reduced time is due to fewer commands being required when moving addresses.

For more information on IP Aliases in HACMP, see the HACMP manuals:

- ▶ *HACMP v5.3 Administration Guide*, SC23-4862-06
- ▶ *HACMP v5.3 Planning and Installation Guide* SC23-4861-07

## Disk resource failover

During a failover, HACMP moves shared disks defined in the resource group from one server to another. This includes the following activities:

- ▶ *Varyon* (activate) VG - **varyonvg** command
- ▶ Mount FSs (if the FSs are defined in the resource group)

The delay in a DB2 failover mostly stems from the necessity to move control of the shared disks and resources from one server to another. An integrity check of file systems (**fsck**) consumes a tremendous amount of time when the shared VG contains huge FSs. The more FSs in the resource group, the longer it takes to failover. Therefore, the basic way to speed up disk resource takeover is to reduce the amount of FSs in the resource group. This can be accomplished by using raw device containers instead of FSs. With DB2 database managed space (DMS) on raw device, we can significantly decrease the disk resource failover time, because it is not necessary to mount and check consistency of FSs during failover.

We can also have a *concurrent access resource group* for the raw disk, reducing failover time even further. *Concurrent access volume group* with HACMP enables both nodes to activate a VG simultaneously. This configuration reduces time for takeover resources, because a shared disk is then always activated on standby nodes and it is not necessary to activate/varyon the VGs either. Using concurrent access with HACMP requires the installation of an additional HACMP file set. For example:

```
cluster.es.clv.m.rte for 5.3.0.0 COMMITTED ES for AIX Concurrent  
Access
```

Concurrent access mode is not supported for JFSs. Instead, you must use only raw logical volumes or physical disks for concurrent access resource groups.

Another consideration for the concurrent access resource group concerns the logistics of ensuring that no processes on the standby node (for example, DB2 instances) are actively using any shared disks. Concurrent VGs are activated

from all the nodes in the cluster. This means that they can be accessible from all nodes in the cluster simultaneously and there is a very real possibility of unintentional data corruption or loss. In this situation, software with data on a shared disk has the responsibility to ensure consistent data access, whether that be performed from within the software, or controlled externally via the application start/stop scripts.

While DB2 databases will not tolerate concurrent access from multiple servers, in mixed/hybrid HACMP/HADR mode, as discussed in Chapter 11, “HADR with clustering software” on page 345, DB2 databases use a shared disk on the primary cluster pair, and HADR writes to the standby on a third server node. Of course, only one DB2 instance can actively use this shared disk at once.

To improve performance of failover/takeover, using a raw device for data storage is recommended as mentioned previously. Then the next question is a matter of what is created on the raw device and what is not.

▶ User table spaces:

If your database is huge, user table spaces should take up most part of the shared disks, so using DMS raw device for the table space container will improve failover performance.

▶ Temporary table spaces:

You can configure temporary table spaces in raw devices too. For huge sequential read and write I/Os, DMS temporary table space is preferable in terms of performance. On the other hand, SMS table space is recommended from the point of view of storage utilization and ease of manageability. One concession to help with DMS containers is the automatic table space resizing feature for table spaces available from DB2 V8 FixPak 9. For details of the automatic table space resizing, see the DB2 InfoCenter:

<http://publib.boulder.ibm.com/infocenter/db21uw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/admin/c0012277.htm>

▶ Catalog table space:

You can specify the characteristics of the DB2 system catalog table space when you create databases with the CREATE DATABASE command. Because the DB2 SYSCATSPACE is small in most cases, it is unlikely to contribute to reduction of failover time.

▶ Active logs:

An active log can be configured in a raw device. However, note that from DB2 version 9, database logging using raw devices is deprecated. For details, see the DB2 Information Center topic “Database logging using raw devices is deprecated”:



<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.rn.doc/doc/c0023086.htm>

► Other database components:

Other database components, such as database directory files, instance home and subdirectories, and archive log directories have to be created on FSs. Note that the archive logs also have the option to use TSM and avoid the issue altogether.

Because we still have FSs for some database components, HACMP fast disk takeover can help to failover these file system resources. This recent feature from HACMP V5.1 enables faster resource group takeover using AIX Enhanced Concurrent Volume Groups (ECM). Enhanced concurrent VG of AIX supports active and passive mode varyon, can be included in a non-concurrent resource group, and the fast disk takeover is set up automatically by the HACMP software.

For all shared VGs created in enhanced concurrent mode that contain FSs, HACMP will activate the fast disk takeover feature. When HACMP starts, all nodes in a resource group sharing the same enhanced volume group will varyon this VG in passive mode. When the resource group is brought online, the node that acquires the resources will varyon the VG in active mode. This reduces the time to varyon VGs, even though it still requires time to mount FSs. The prerequisites for this functionality are:

- HACMP V5.1 or later
- AIX 5L™ 5.2 or higher
- bos.clvm.5.2.0.11 or higher
- APAR IY44237

For more information about fast disk takeover, see *HACMP for AIX 5L V5.3 Planning and Installation Guide*, SC23-4861-06.

There is another HACMP option to speed up mounting FSs: parallel mount of FSs. This feature allows for specification of how FSs are mounted during disk resource takeover. Parallel mount can be faster than sequential (default). This option can be used if you do not have nested FSs.

You can also change the method of the file system integrity check from the default `fsck` to `logredo`. Although `logredo` can speed up mounting FSs, this option does not guarantee that all the errors in the FSs can be fixed if the node crashes in the middle of file system I/O.

## Starting and activating the database

After the resources are taken over by the standby node, the next thing to do is to re-enable the DB2 database for client use, which includes the following activities:

- ▶ Restart the DB2 instance (by the **db2start** command)
- ▶ Activate the database, including buffer pool activation and crash recovery

If crash recovery is required, it can consume some time before the database can be opened up for client access. From the perspective of high availability, it is imperative to reduce crash recovery time and restart the database more rapidly.

Crash recovery is the process by which the database is moved back to a consistent and usable state. This is done by rolling back incomplete transactions and completing committed transactions that were still in memory when the crash occurred. In other words, crash recovery updates the data in table space containers to match what is stored in any unprocessed log records. This means that time consumed by crash recovery depends on how large the gap is between the log records and the actual data in database containers. Here, the gap equates to the amount of log records, as follows:

- ▶ Records to be rolled forward: The amount of log records that have been committed, but have not been written from the buffer pool to the disk.
- ▶ Records to be rolled back: The amount of log records that have not been committed, but have been written to disk.

To make crash recovery fast, you have to reduce these amounts:

- ▶ Records to be rolled forward:

You can synchronize memory and database pages more frequently by writing the dirty pages on buffer pool to the database pages. We introduce some relevant tuning parameters next.

- NUM\_IOCLEANERS:

The page cleaner DB2 process (I/O cleaner) is in charge of writing dirty pages on the buffer pool to the database pages on disk. The number of these processes activated is determined by the database configuration parameter NUM\_IOCLEANERS. When you increase this parameter, the contents of the database are updated more on disk, because the page cleaner cleans dirty pages from buffer pools to data pages and therefore this will reduce the crash recovery time. We recommend that you increase this parameter depending on the number of the CPU cores on the server.

- SOFTMAX and CHNGPGS\_THRESH:

The page cleaners are started according to the setting of the database configuration parameters SOFTMAX and CHNGPGS\_THRESH. Tuning these parameters lower than default values helps to reduce crash recovery time, with the trade-off of increasing the load on certain system resources.

When the SOFTMAX parameter is set lower, it will cause the database manager to trigger the page cleaners more often and take more frequent

soft checkpoints. The log control file is then written to disk more often and thus can reduce crash recovery time.

The lower that the CHNGPGS\_THRESH parameter is set, the lower the percentage of changed pages is required to be kept in the buffer pool. This will trigger the asynchronous page cleaners to start cleaning the buffer pool more often. This also means it writes committed data to the disk more often and therefore less recovery time is needed in case of crash recovery.

While in theory, spreading out smaller but more frequent writes to disk from buffer pool pages should reduce peak write I/O load, and reduce the frequency of synchronous write I/O, frequent writes to storage by page cleaners may still sometimes lead to adverse performance impacts. Consider the performance impacts to system load and transaction throughput and tune for the best values to suit your system.

From DB2 V8.1.4 onwards, consider using registry variable DB2\_USE\_ALTERNATE\_PAGE\_CLEANING for even further improved and proactive page cleaner activity. This registry variable makes CHNGPGS\_THRESH redundant, as it is no longer controls page cleaning.

► Records to be rolled back:

It is important to frequently issue commit from the applications. This cannot be stressed enough; issues of transactional performance, concurrency, and ease of recoverability are heavily dependent on how applications are coded.

Despite giving the appearance of a continuous connection to a database back-end, a well coded online transaction will perform most of its work without even holding locks on data, and will only need to gain locks momentarily before immediately committing in order to process requests after the user has confirmed a given action.

Batch processing is also a frequently abused target from users who expect to be able to issue logged updates of millions of rows at a time, and then get upset when a logical failure or timeout occurs halfway through their input files, and their transaction then takes time to rollback. These are often the same users who complain when told that the answer is to commit as frequently as possible, meaning that they will have to code retry and restart logic into their batch jobs, or pursue alternatives such as unlogged loads with copy yes options.





## DB2 with Microsoft Windows Server Cluster

In this chapter we provide the steps to set up a highly available DB2 environment with Microsoft Windows Server Cluster. We explain the fundamental concepts of Microsoft Windows Server Cluster to help you understand its architecture and the basic configuration procedures in Windows 2003 Advanced Server.

We discuss how to configure a DB2 instance and other cluster resources to become the highly available resources in Windows Server Cluster.

We demonstrate the configuration steps using both the automatic utility db2msc and the manual steps.

**Note:** This chapter uses the screen captures from Microsoft Windows Cluster. Microsoft product screen captures are reprinted with permission from Microsoft Corporation.

## 10.1 Server Cluster concepts

Microsoft's answer to high availability requirements in business is Windows Server Cluster, formerly known as Microsoft Cluster Servers (MSCS).

MSCS was introduced early in the Windows NT® 4.0 Enterprise edition as a separate component from the operating system.

With the introduction of Windows 2003 Advanced Server, Enterprise Edition, and Data Center editions, MSCS has been revamped in what is now called Windows Server Cluster (which we refer to as Server Cluster hereafter). Server Cluster is an integral part of the operating system and therefore cannot be uninstalled. It is always present (under %windir%\Cluster), therefore, no additional software pieces have to be installed.

The objective of this chapter is not to give an exhaustive description of Server Cluster but rather to introduce the fundamental concepts so that the DB2 DBA can consider this feature with DB2 when selecting a clustering product for a DB2 application.

In this section, we introduce some concepts particular to Server Cluster. Some general concepts from cluster technologies are not mentioned because they have been described in Chapter 1, "DB2 high availability and disaster recovery options" on page 1.

### 10.1.1 Types of clusters

Server Cluster provides technologies to create two types of highly available servers, depending on what services must be made available:

- ▶ Network Load Balance (NLB)
- ▶ Quorum server cluster

#### **Network Load Balance**

Network Load Balance (NLB) is a type of cluster that allows enterprises to create a farm of servers (nodes) that provide a service to applications. In the event of a failure in one of the nodes, the workload of that node can be assimilated by the rest of the nodes in the farm. This type of cluster is suitable for applications that do not store a state. Applications that are usually capable to scale-out horizontally, such as Proxy servers, Web servers, and so on, are the target for this type of clustering solution.

An NLB server uses algorithms to balance the load of each node to generate the best possible overall performance. NLB is included in all versions of Windows 2003.

### **Quorum server clusters**

Quorum server clusters can be used for applications that store a state (such as a Database Management System) and are used for services that usually scale vertically and have to be made highly available.

Because these services can scale-out vertically, a server farm cannot be created, and the only way to make the server highly available in case of node failure is to have a spare machine that will take over the functions of the failed node.

Server Cluster requires the use of special shared disks among the nodes or the use of complex algorithms among nodes to synchronize the state of the nodes and the services they provide.

Quorum server clusters come in two different types:

- ▶ Single quorum device cluster
- ▶ Majority node set (MNS) cluster

#### ***Single quorum device cluster***

Single quorum device clusters use a shared disk device known as quorum. These shared disks are SCSI disks in a shared bus or a Storage Area Network (SAN) using Fiber Optics.

One of these shared disks is known as the *quorum* for the cluster. This quorum is used as an arbitrator to track which node is the owner of the resources and services. If the node that owns the resource goes down, Server Cluster takes action to move the service to another node.

#### ***Majority node set cluster***

MNS consists of two or more nodes that do not have a single shared quorum. Each node in the system has their own copy of the quorum in a local disk. The Server Cluster uses complicated algorithms to keep the local quorums synchronized. These nodes may or may not be attached to one or more cluster storage devices.

MNS clusters are more commonly seen in a geographically dispersed cluster configuration and that is the reason for not using shared devices. The down side of this type of configuration is that for the cluster to work, you require half of the nodes plus one online and interconnected.

This rule is necessary because all clusters are connected only throughout the network. The only way to know if a server has the correct quorum information is by knowing that the *majority* of the nodes have the same information.

## 10.1.2 Windows Server Cluster definitions

From now on we refer only to single quorum device clusters. In this section, we introduce a few concepts that are necessary to understand how single quorum device clusters work in Windows Server Clusters.

### Resource type

In Server Cluster, a resource type is a service, such as Web service, IP address, or a device, such as a shared disk, that has to be made highly available.

For every type of resource to be manipulated, you require a handler which is an interface that will manipulate this resource. For example, if what you want to make highly available is a shared disk, you require a mechanism to:

- ▶ Detect that the resource is working properly.
- ▶ Detect the availability of the disk from each node.
- ▶ Failover or failback the resource among the nodes as necessary.

In the case of Server Cluster, this handler is implemented as a Dynamic Link Library (dll) that must be provided by the provider of the service or resource. In the case of basic operating systems or resources (such as disks), Server Cluster provides a default dll that can handle them.

In the case of DB2, a new type of resource known as DB2 should be created. 10.7.1, “Adding the DB2 resource type” on page 329 shows how to add DB2 resource type to Server Cluster.

### Resource

A resource is any element that you want to make highly available in the system. Common resources that you will find are:

- ▶ IP addresses
- ▶ DB2 instances
- ▶ File shares, which are shared directories in Windows
- ▶ Shared disks



## Groups

A cluster group is a set of resources that are interdependent of each other. They are logically linked together for a specific purpose, for example, to make a DB2 instance highly available.

Groups also make it possible to establish ownership of the resources to each node. When the group is created, you must specify what nodes can use the resources in that group and which node is the initial owner.

## 10.2 Minimum configuration and sample setup

In order to set up a single quorum Server Cluster using Windows 2003, you require at the very least:

- ▶ A Windows 2003 domain.
- ▶ Two machines to work as the nodes in the cluster using the Windows 2003 editions that support the Server Cluster. For more information about the latest Windows 2003 editions supporting the Server Cluster, refer to the Microsoft Web site:

<http://www.microsoft.com/windowsserver2003/default.mspx>

- ▶ A shared disk that can be accessed simultaneously from all the nodes.
- ▶ Two network cards in each node (desirable but not mandatory).

In our lab example, we have the following resources:

- ▶ We set up a machine Wisla, a Windows 2003 standard server, working as the Domain Controller for domain CHAVIN.
- ▶ Machines Mochica and Chimu are running Windows 2003 Advanced Server.
- ▶ Each node has a QLogic® QLA2200 PCI Fiber Channel adapter to access the SAN devices.
- ▶ Four disks are created in the SAN infrastructure for the nodes:
  - Drive Q: This has 1 GB space and has been created to be the quorum of the cluster.
  - Driver R, S, and T: These are 50 GB partitions for DB2 table spaces and instance profile storage.

Following is the procedure to set up a highly available DB2 environment with Server Cluster. We provide detailed steps for creating Windows Server Cluster and configure DB2 with it in the next few sections. Refer to the Windows documentation for the steps mentioning the Windows domain controller.

1. Install Windows 2003 editions supporting Server Cluster on each machine that is a cluster node.
2. Configure the shared disk devices and make sure that each node has access to the shared resources.
3. Add the nodes to the Windows Domain.
4. Create a Server Cluster.
5. Add nodes to the Server Cluster.
6. Install DB2.
7. Create a DB2 instance.
8. Make the DB2 instance highly available.

## 10.3 Creating a Server Cluster

In order to create a cluster, you require a Windows Domain. All the information related to the cluster is stored in the active directory of the domain. The machines that are part of the cluster must use Windows 2003 Enterprise Edition or Windows 2003 Data Center Edition. The cluster services are not available in Windows 2003 standard edition.

We follow these steps to create a Server Cluster:

- ▶ Add the machines that are the nodes to the domain.
- ▶ Create the cluster in the domain.

### 10.3.1 Adding the machines to the domain

The first step is to attach the machines that will conform the cluster into the Windows domain. In order to accomplish this, follow this procedure for each machine.

1. Use Administrator user ID, go to **Start** → **Control Panel** → **System**. In System Property, select the **Computer Name** tab, then click **Change**. See Figure 10-1.

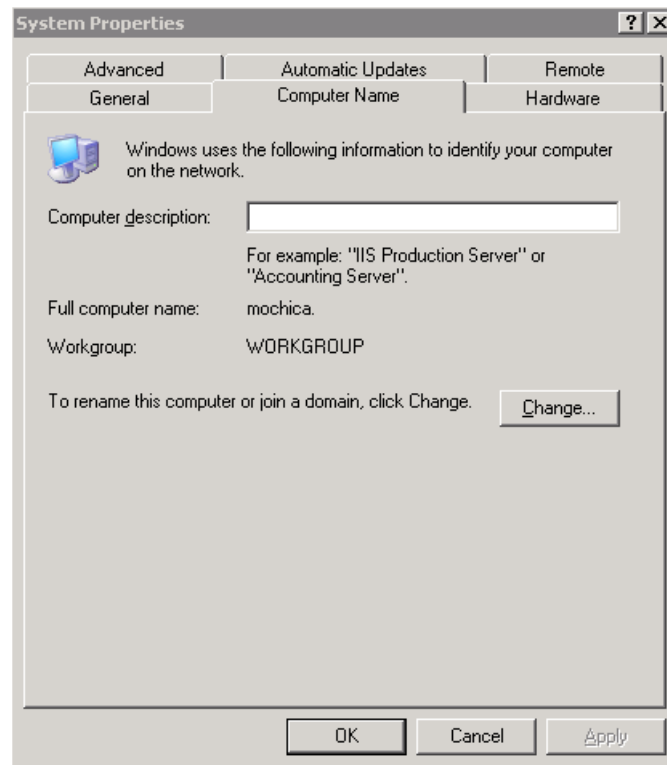


Figure 10-1 System Properties

2. In the Computer Name Changes window, select **Domain** and enter the Windows Domain name you want to join, CHAVIN in this case. Click **OK**. See Figure 10-2.

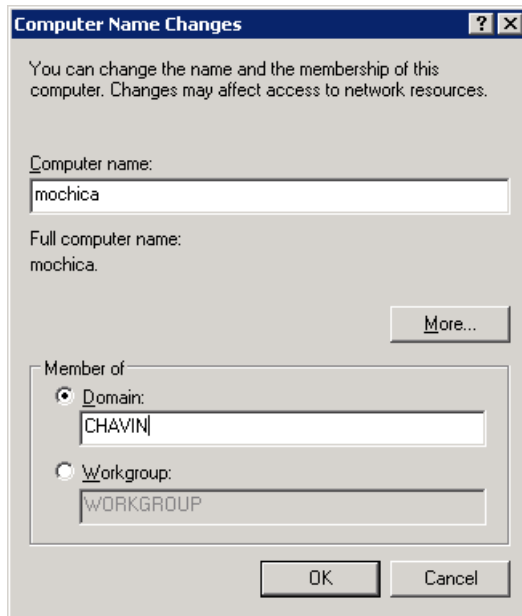


Figure 10-2 Add computer to the domain

3. A message is shown indicating that the machine was accepted on the domain. See Figure 10-3.

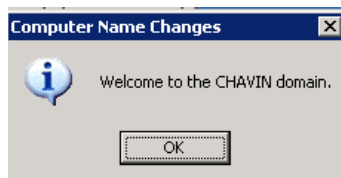


Figure 10-3 Machine accepted message

## 10.3.2 Creating the cluster in the domain

The following steps are for creating a cluster in the domain. Remember that in order to create a cluster, a shared disk resource must exist. This shared disk must be connected to both nodes using a SCSI shared bus or a SAN adapter.

1. Go to **Start** → **Administrative Tools** → **Cluster Administrator**. The Cluster Administrator opens the Create New Cluster option (Figure 10-4). Click **OK**.

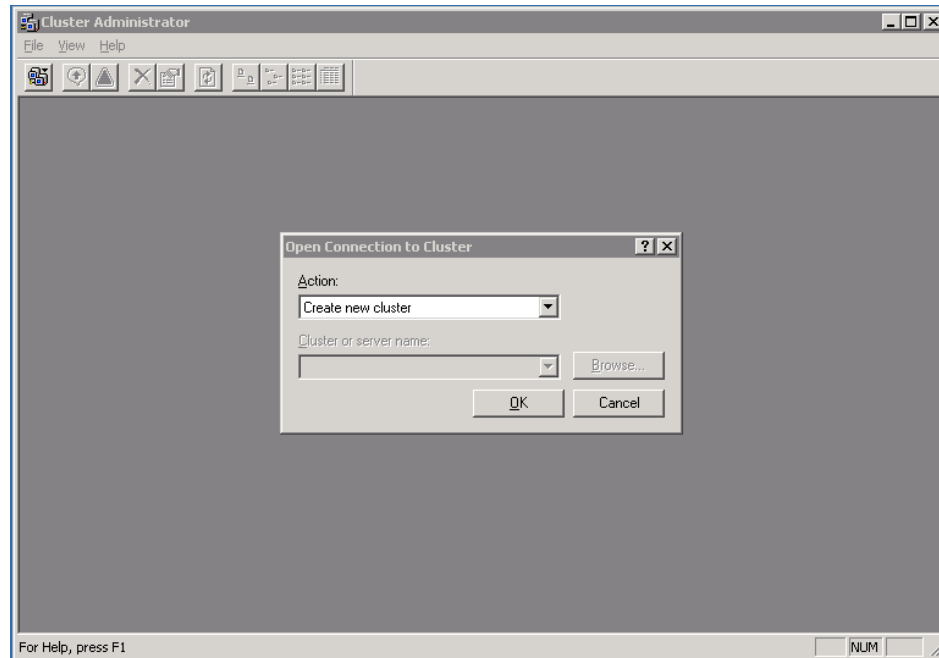


Figure 10-4 Creating a new cluster

2. The Create New Server Cluster Wizard is shown in Figure 10-5. Click **Next**.



Figure 10-5 Create New Server Cluster Wizard

3. In the Cluster Name and Domain window, select the domain where the cluster is created and choose a new name for the cluster. This name is registered in the domain and linked to a highly available IP address. See Figure 10-6.

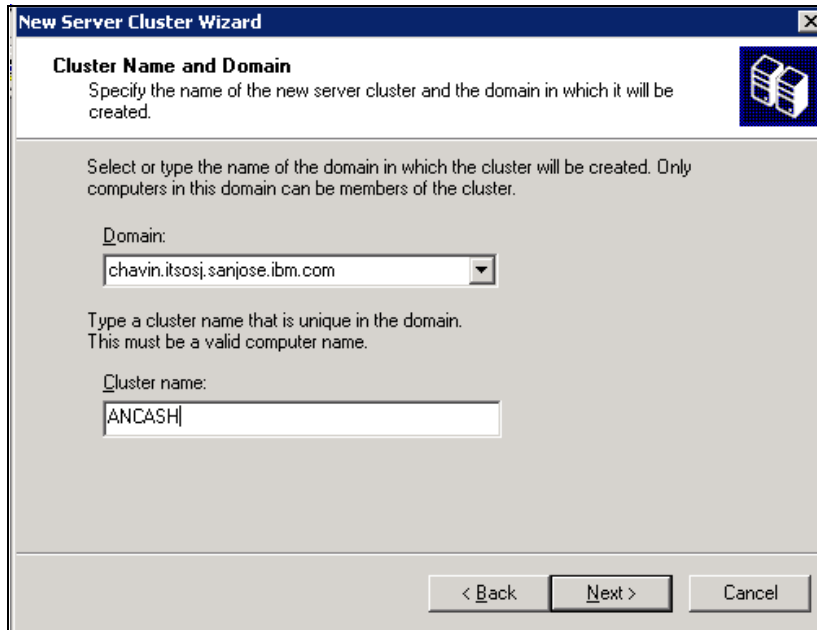


Figure 10-6 New Server Cluster Wizard

4. Select the Computer window shown, enter the computer name and click **Advanced**.
5. In the Advanced Configuration Options window (Figure 10-7), you can select either configuration. In this example, we chose **Advanced (minimum) configuration** to configure every component.

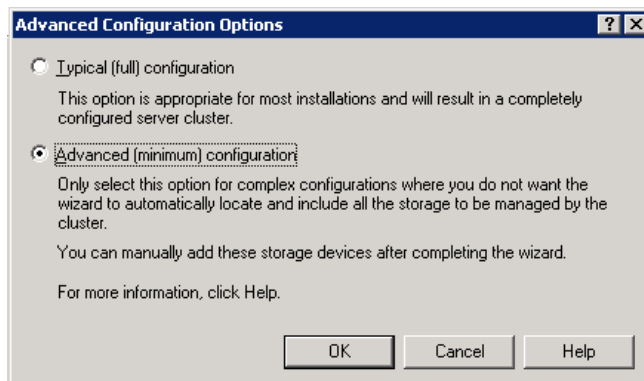


Figure 10-7 Advanced Configuration Options

6. If you have more than one shared disk, click **Quorum** in the Proposed Cluster Configuration window to select the disk drive as the Quorum device, or click **Next**. See Figure 10-8.

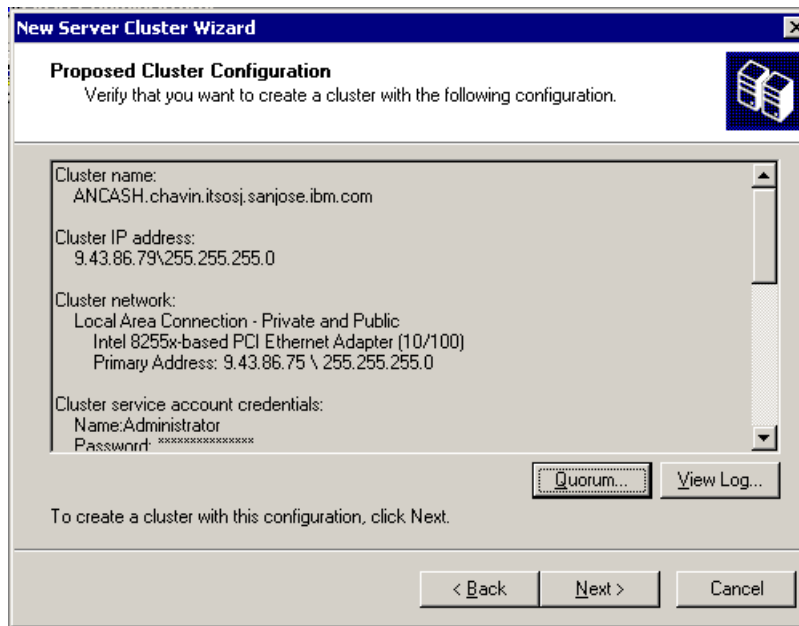


Figure 10-8 Proposed Cluster Configuration

7. If you click **Quorum** in the Proposed Cluster Configuration window to set up the Quorum, a Cluster Configuration Quorum window pops up. See Figure 10-9. In this window, choose one to be the Quorum device for this cluster. In our example, we chose the Q drive.



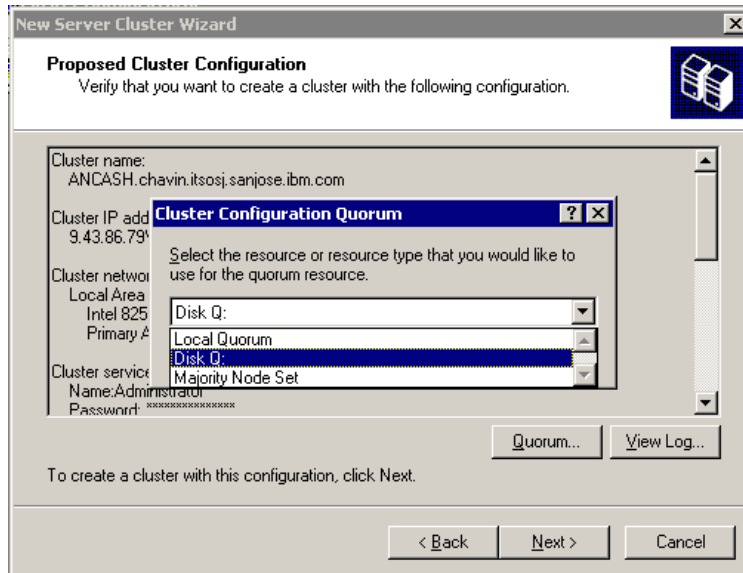


Figure 10-9 Choose the Quorum device

8. Now you have set up a cluster with one node. See Figure 10-10.

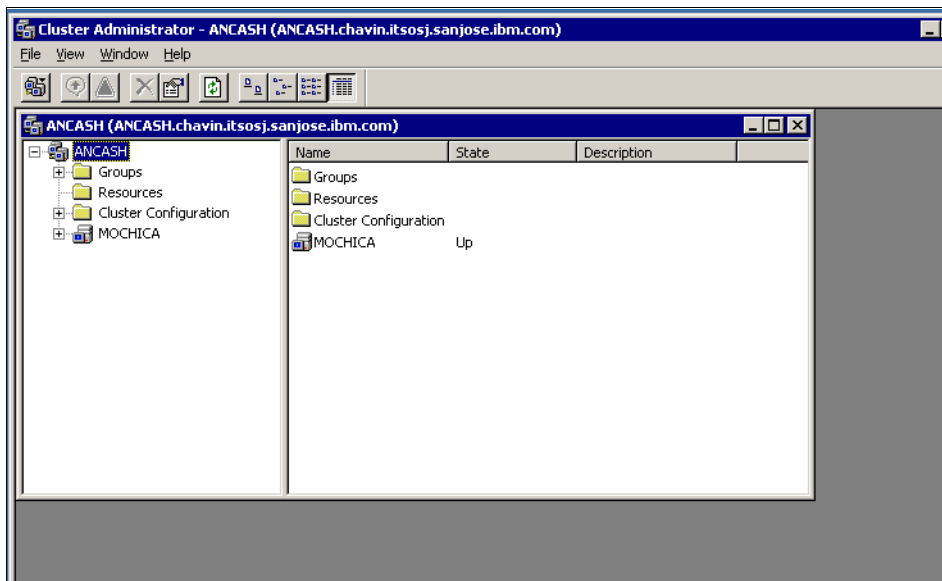


Figure 10-10 Cluster is set up with one node

9. From Windows Explorer, expand the Quorum drive. If the cluster is set up successfully, the MSCS directory is created and the directory contains the quorum log file quolog.log and a temporary file. See Figure 10-11.

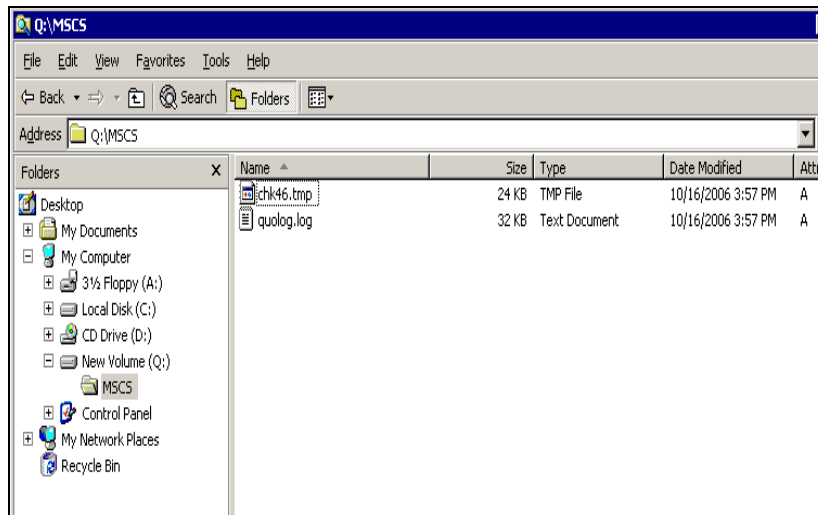


Figure 10-11 MSCS directory

The next step is to add the second node using the following steps:

1. To add a new node, connect to the cluster from **Start** → **Administrative Tools** → **Cluster Administrator**. In the Open Connection to Cluster window, enter the cluster name just created and click **OK**.
2. In the Cluster Administrator for the cluster window, select **File** → **New** → **Node**. The Add Node Wizard welcome window is shown. Click **Next**.

3. In the Select Computer window, enter the machine name to be added to the cluster in the Computer Name field and click **Add**. The machine name should be shown in the Selected computer area. See Figure 10-12.

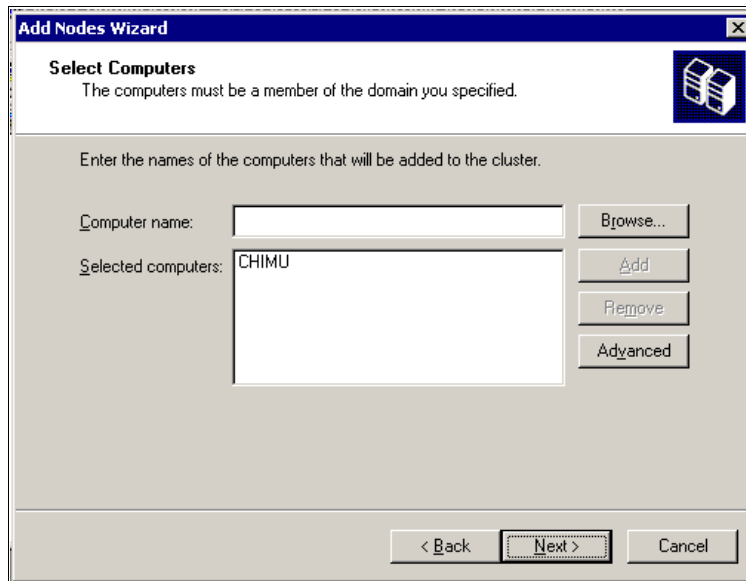


Figure 10-12 Enter new node name

4. The Add Node Wizard analyzes if it is possible to add the node by checking the available resources. In this case, there is a warning message as the machine we used had only one network adapter and two is the recommended configuration. See Figure 10-13. Click **Next**.

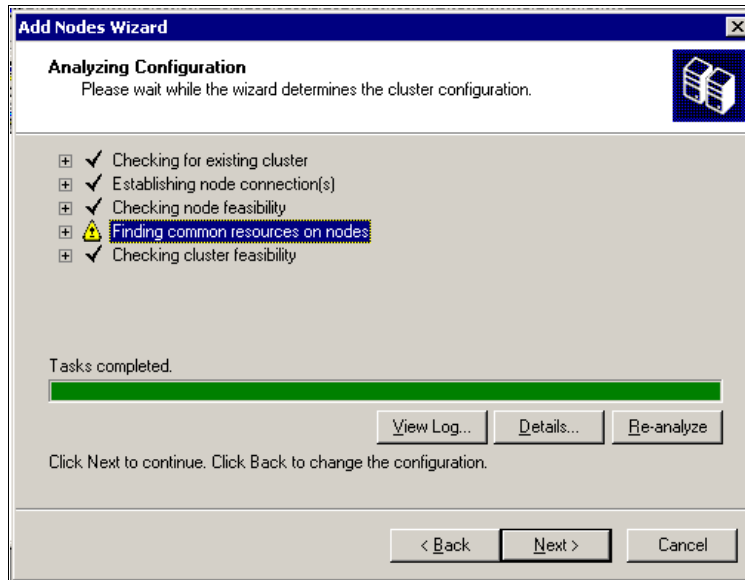


Figure 10-13 Adding a new node

5. The Administrator user and password on the domain are required to complete the operation (Figure 10-14). After this step, the new node should be added to the cluster.

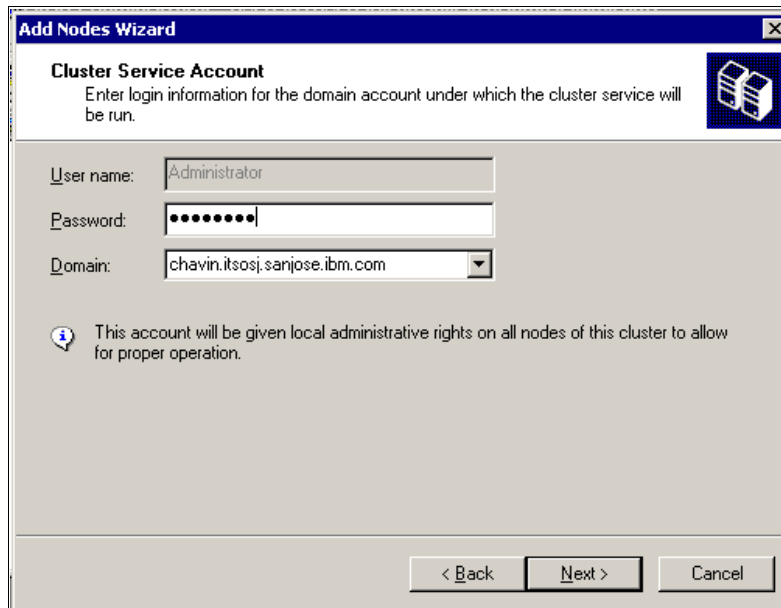


Figure 10-14 Cluster Service Account

When the setup steps are completed in both nodes, verify that the files `%windir%\cluster\CLUSDB` and `%windir%\cluster\CLUSDB.log` exist in both machines. These are the database and the log file for the cluster respectively.

## 10.4 Installing DB2 ESE

DB2 should be installed on all the nodes in the cluster. For the DB2 installation procedure, refer to DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.server.doc/doc/t0008099.html>

DB2 creates a default instance along with the DB2 installation. However, we create a new instance to configure the DB2 cluster in this example.

## 10.5 Adding resources to a cluster for DB2

Initially, in the cluster, you will have a single group called Cluster Group that contains the following resources:

- ▶ Cluster IP address
- ▶ Cluster name
- ▶ Quorum drive

In our example, the highly available cluster IP address is set to 9.43.86.169 and the quorum device is Disk Q.

We have to add the shared disks as a resource for the cluster so the disks can be used to store the instance profile (we use Disk R: in our example).

Use the Server Cluster adding resource to cluster steps to add the shared disk as follows:

1. In the Cluster Administrator utility, go to **File** → **New** → **Resource**.
2. In the New Resource window (Figure 10-15), enter:
  - a. Name: Disk R
  - b. Resource Type: Select **Physical Disk**. Then click **Next**.

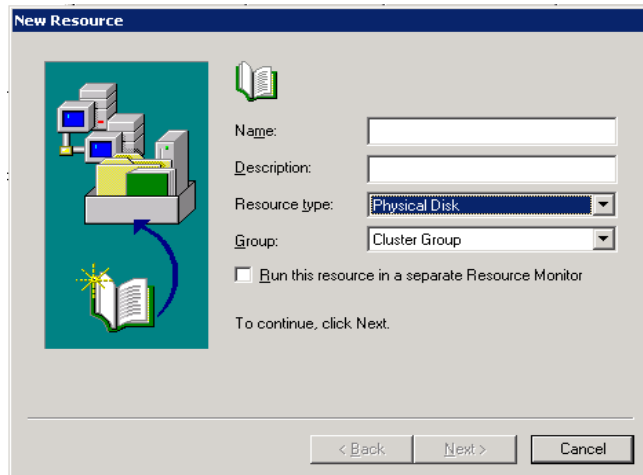


Figure 10-15 Add a new resource

3. The Possible Owners dialog is shown in Figure 10-16. By default, all nodes in the group are selected. The order is not important, because the owner of the resource becomes the group owner. Click **Next**.

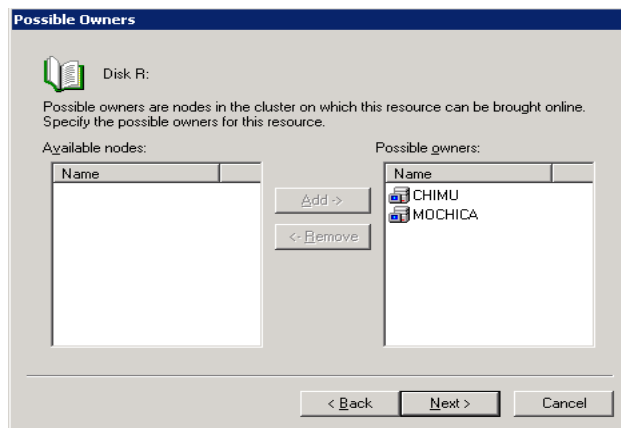


Figure 10-16 Possible Owners

4. You are presented with the Resource Dependency dialog. If the resource that you are adding requires any other resource in order to work properly, you must add that resource in this dialog. For example, if the resource that you are adding is a DB2 instance, you will have to add the disks on which the DB2 instance is created to the dependency list. This way, you will avoid the possibility that DB2 is started before the disks are ready.

In this example, we are adding a physical disk that does not depend on any other resource. Therefore, the dependency list is empty (Figure 10-17).

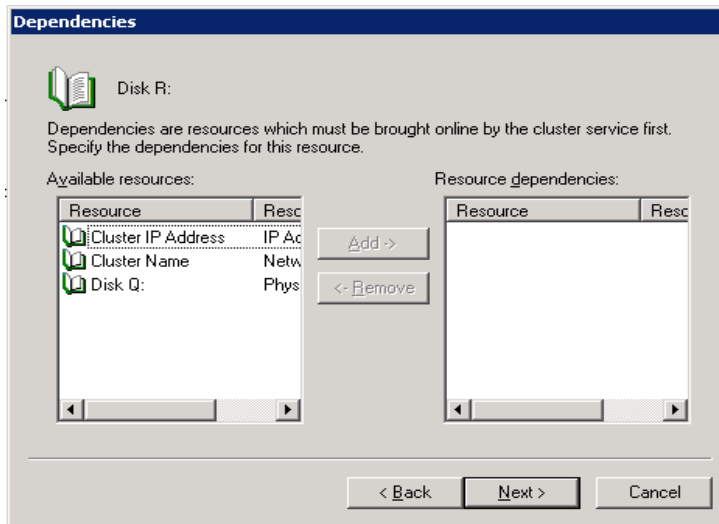


Figure 10-17 Resource Dependency

5. The last step to add a resource is to specify the resource-specific parameters. For example, if you are creating a highly available IP address, in the last step you are asked for the specific IP address you want to use and the netmask.

Because we are adding a physical disk, the last step is a drop-down list box with all the disks available in the system (Figure 10-18). In this case, it is Physical Disk R.

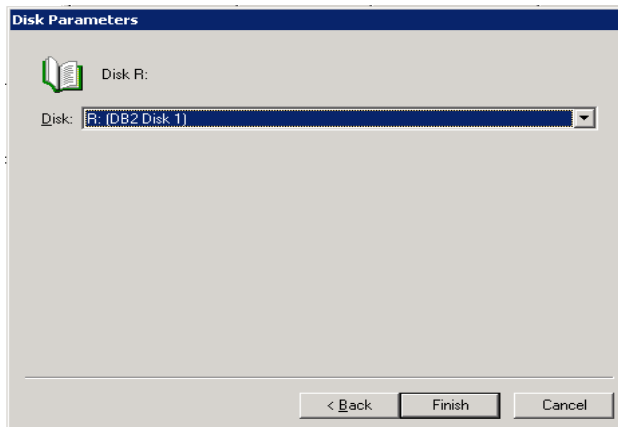


Figure 10-18 Resource Parameters



From the Cluster Administrator, we can see that the physical disk has been added successfully as shown in Figure 10-19.

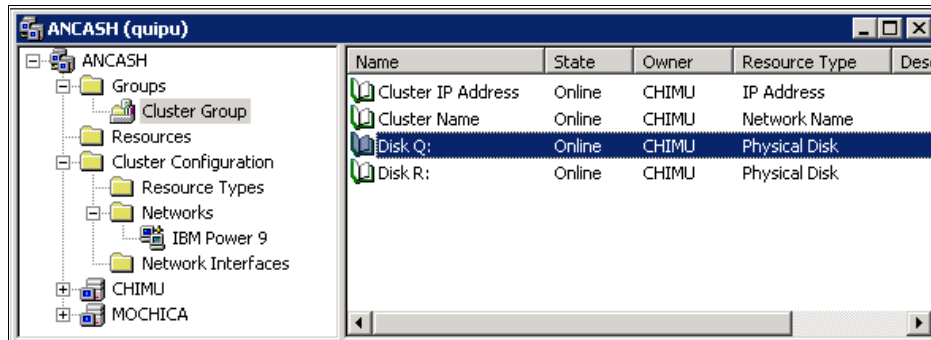


Figure 10-19 After adding a physical disk

## 10.6 Creating a DB2 instance

We create a single partition ESE instance using the `db2icrt` utility in the node to be used as the initial resource owner in the cluster. In our example, we created a DB2 instance in Mochica. The instance name is `DB2PERU` and the owner of the instance is the Domain user `db2admin`.

Figure 10-20 shows the command used to create the instance and the result of the command. Note that `db2icrt` will request the password of the domain user ID specified in the `-u` option.

The second command `db2ilist` in the same screen lists the instances in the local machine. Use the `-w` flag in the `db2icrt` command to set the type of instance to a Workgroup server or Enterprise server edition. Additionally, you can use the `-p` option to assign a port to the instance if necessary.

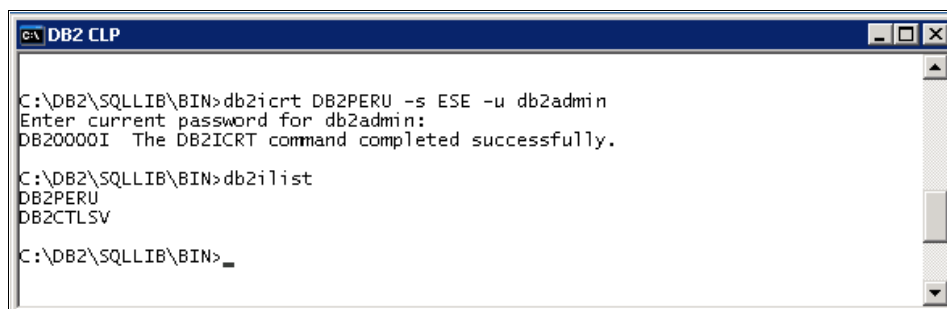


Figure 10-20 DB2 instance creation

If you open the Windows Services (**Start** → **Administrative Tools** → **Services**), you should see a new created service that represents the instance.

In the case of an ESE instance, the name of the service has the suffix `<INSTANCE-NAME>-<NODE NUMBER>`, for example, `DB2-DB2PERU-0`.

In the case of a WSE instance, the name of the service has suffix `<INSTANCE-NAME>`. If we created a WSE instance, the service is like `DB2-DB2PERU`.

This name is very important, as it becomes the Windows Registry Key Name of the service. This means that the following entries must exist in the Windows registry:

▶ ESE instance:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DB2PERU-0
```

▶ WSE instance:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DB2PERU
```

The cluster software looks to this entry to start a highly available DB2 instance. This is usually the most common problem in setting up DB2 with Windows Cluster. If you name the DB2 resource with a name that is different from this entry, the resource will *not* find the DB2 server and will not start. Renaming the resource will not solve the problem and you will have to recreate it.

In the following sections, we discuss two different methods to make the DB2 instance highly available with Windows Cluster: the manual method and the automatic method using `db2mscs`. Note that we only show a hot standby configuration for a single partition database here. For more information about mutual takeover and multiple partition configuration, refer to the DB2 Information Center:

<https://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.1uw.admin.ha.doc/doc/c0007402.html>

## 10.7 Manually configuring a DB2 instance in Windows Cluster

Although there is a semi-automatic utility `db2mscs` that does most of the work for you, it is important to understand what steps are involved in making a DB2 instance highly available in a Windows Server Cluster environment, so that you are able to diagnose any problem that occurs.

The following procedure can be used to configure a DB2 instance in a Windows cluster to provide high availability:

- ▶ Add a DB2 resource type to the cluster.
- ▶ Create a Cluster group to group all the resources for the instance.
- ▶ Create or move the supporting resources to the DB2 group:
  - A highly available IP address is mandatory.
  - A highly available shared disk to store an instance profile is mandatory.
  - If this is a multi-partitioned database, the instance profile directory must be shared and made highly available.
- ▶ Move the DB2 instance to the cluster.
- ▶ Add the instance to the remaining nodes.

### 10.7.1 Adding the DB2 resource type

After DB2 is installed and the instance is created, you have to create a DB2 resource type in Server Cluster. A resource type is basically a dynamic load library that is registered in Windows Server Cluster to manage a specific type or types of resource. To allow Windows Server Cluster to manage the DB2 instance, the DB2 instance must be a resource in the cluster.

In the Cluster Administrator, under Cluster Configuration, the Resource Type folder contains the current resource types that can be managed by the Server Cluster and the DLL that manages them. See Figure 10-21.

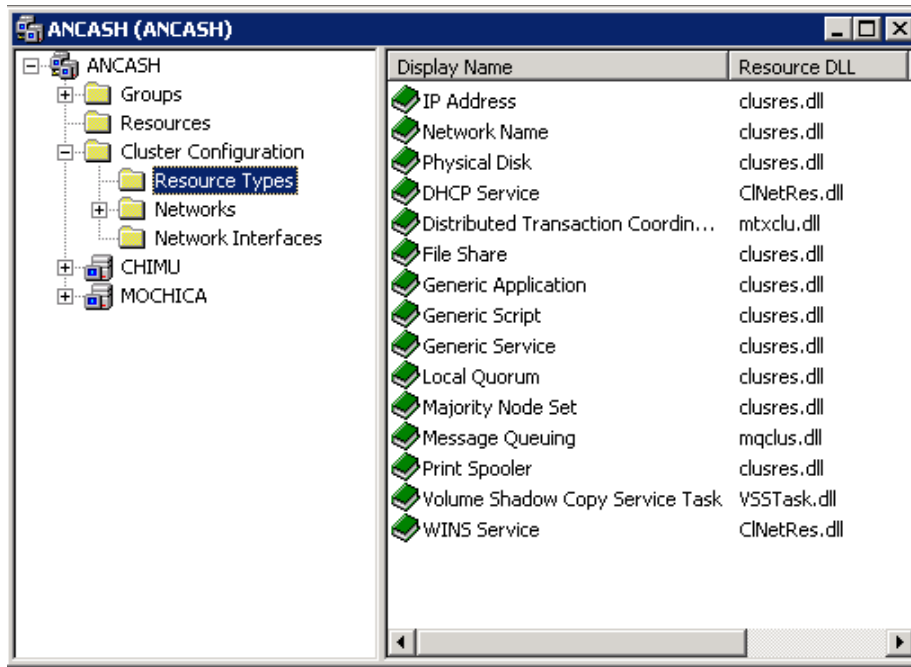


Figure 10-21 Initial resource list in Cluster Administrator

The DB2 resource type can be added to this list using the *db2wolfi* utility. This utility accepts one parameter:

**db2wolfi u | i**

Where:

- ▶ **u**: Unregister the DB2 resource type
- ▶ **i**: Installs the resource type

Figure 10-22 shows the registration of the DB2 resource type. If you try to register a resource that was already registered, you will get the error code 183.

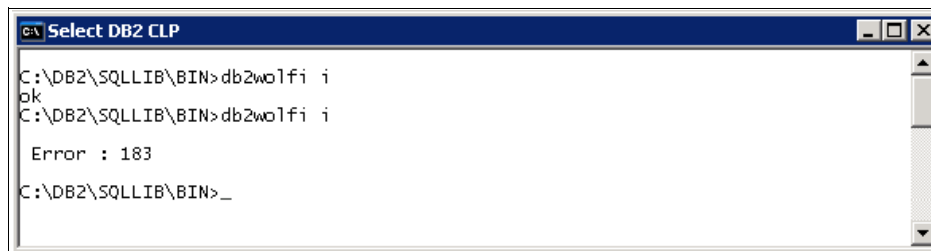


Figure 10-22 Adding DB2 resource type

After db2wolfi is run successfully, the new resource type DB2 is added. See Figure 10-23.

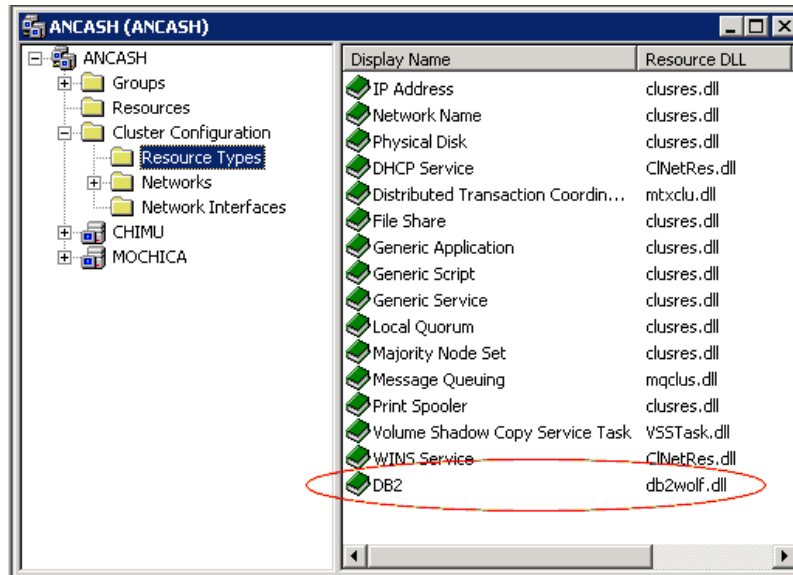


Figure 10-23 DB2 resource type added

After DB2 has been added as one of the resource types, the Windows Server Cluster can manage a DB2 instance to provide highly available service.

## 10.7.2 Creating a DB2 group

In order to create a highly available DB2 resource, we recommend that you place this resource in a separate group that can be failed over, failed back, and made online/offline independently from the main cluster group.

The group can have any name. The recommended naming convention is to name it after the instance. In our example, the DB2 instance is DB2PERU. We create a group called DB2PERU-0 Group.

To create a group in the cluster, follow these steps:

1. From Cluster Administrator utility go to **File** → **Cluster** → **New** → **Group**.
2. The New Group wizard starts. Enter the group name in the Name field. We name this group DB2PERU-0 Group. See Figure 10-24.

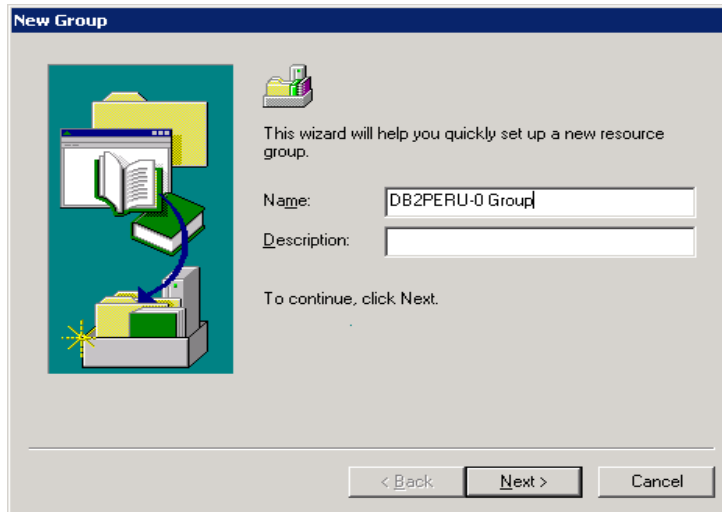


Figure 10-24 New Group wizard

3. In the Preferred Owners window (Figure 10-25), add the nodes that are the owners of this resource.

**Note:** The node sequence makes a difference here. The node at the top of the list in the Preferred Owners list is the initial owner of the group. All resources created inside a group are initially owned by the group owner.

The owner of the resource is the node that can use the resources. In the case of a shared disk, for example, even the disk is shared and both nodes have paths to access it. The cluster software makes it so only the current owner can read and write that device.

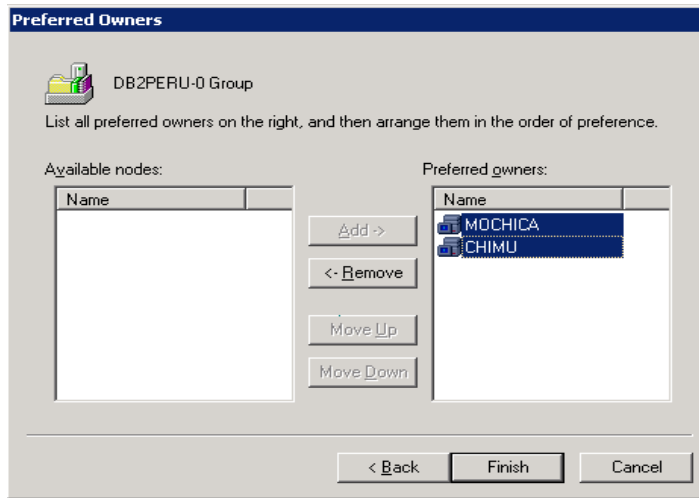


Figure 10-25 Owners

When the group is created, notice that there is a red arrow next to the group name that indicates that the group is offline (not working). This is due to the fact that the group does not have any resource defined and therefore is not working.

You can now move the resource Disk R: created before into the new group. We will use this resource to store the instance profile. You can move the resource to a group by just dragging the resource from the Cluster Group folder and drop it into the new group. Respond **Yes** to confirm the action. When the group contains a resource, the group is up automatically. Figure 10-26 shows that the DB2PERU-0 Group contains resource Disk R.

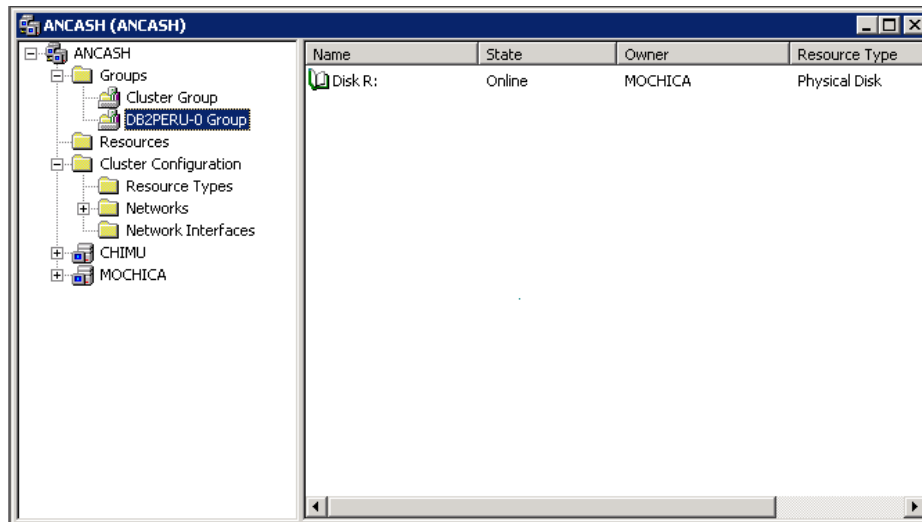


Figure 10-26 DB2 group created

### 10.7.3 Creating a highly available IP address for the DB2 resource

DB2 clients access the DB2 server via TCP/IP. The IP address, which the DB2 client used to communicate with the DB2 server, also has to be included in the Windows Cluster resource group. In the event of a machine failure, the other node will take over the IP address, and clients are able to attempt a reconnect.

To add an IP address to the cluster, follow the steps described in 10.5, “Adding resources to a cluster for DB2” on page 324. Enter the resource name and select **IP Address** as the Resource type as shown in Figure 10-27.



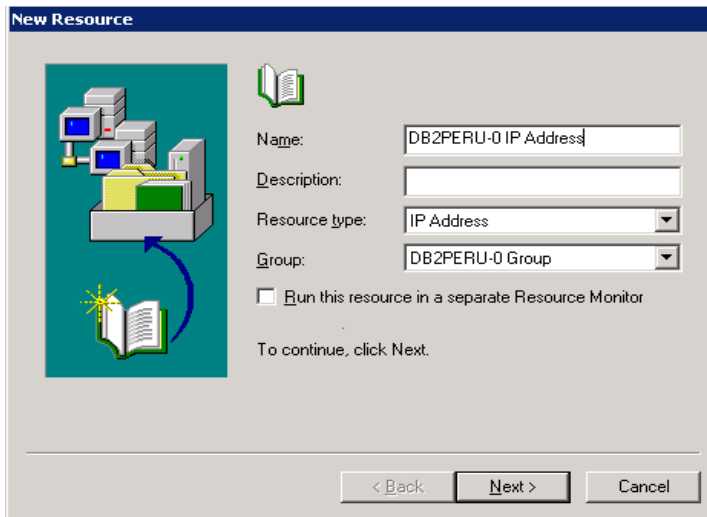


Figure 10-27 Creating IP Address resource

In the last step, you are asked the IP address to use as well as the netmask. See Figure 10-28. After the creation is complete, you will have to bring the resource online by right-clicking the resource and selecting **Bring online**.

After you see that the IP address is online, try to ping it from another machine to ensure that it is working properly.

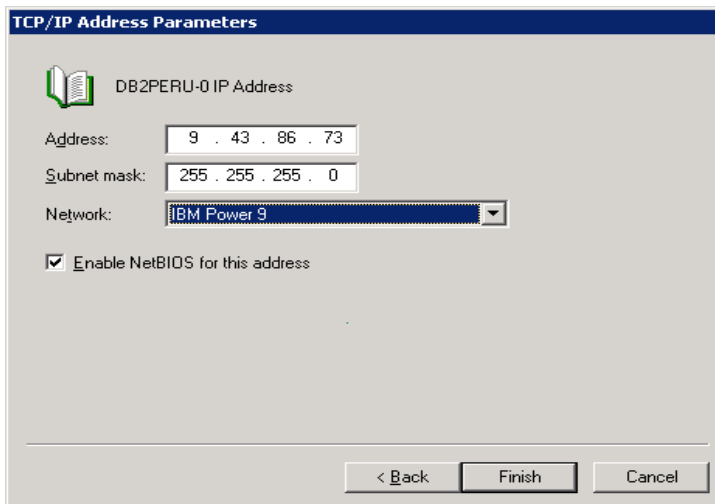


Figure 10-28 New IP Address resource created

## 10.7.4 Migrating the DB2 instance to the cluster environment

The instance, as it is now, has been created in only one node in the cluster. In the event of a failure, the other node would not have been able to find information about this instance. The same applies for any other resource in the cluster. Windows Cluster maintains information about the status, resources, resource types, nodes, and so on, of the cluster in a file located in %WinDir%\Cluster\CLUSTDB. This file exists in each node and is synchronized constantly by the Server Cluster software.

Additionally, the applications in the cluster have to access configuration information in any of the nodes. To avoid replicating potentially multiple registry keys in the registry, most of the contents of the CLUSTDB and all information related to a highly available resource must be located in the same Windows Registry key:

```
HKEY_LOCAL_MACHINE\Cluster
```

This key is also synchronized among the nodes by Windows so applications can use it in any of the nodes.

The process of migrating a DB2 instance to the cluster includes two steps:

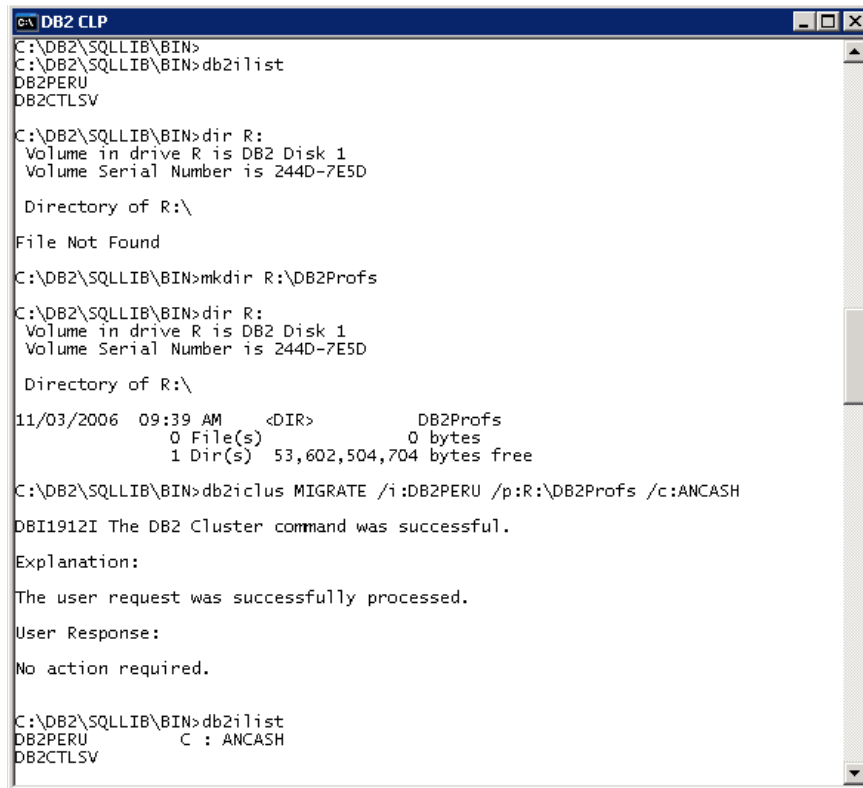
1. Moving the DB2 instance profile directory to a highly available disk where it can be found by any node.
2. Creating a Windows register key in the cluster section of the Windows Registry.

These two tasks are accomplished using the **db2iclus** utility, which must be run in the node where the DB2 instance was created.

In Figure 10-29, you can see a series of commands:

- ▶ **db2ilist** shows you that there are two instances in the system. None of them are available in the cluster.
- ▶ **mkdir R:\DB2Profs**: In the highly available disk Drive R, we create a directory called DB2Profs to store the instance profiles.
- ▶ **db2iclus MIGRATE /i:DB2PERU /p:R:\DB2Profs /c:ANCASH**: The **db2iclus** command moves the instance DB2INCA to the cluster ANCASH and places the DB2 instance profile directory in the directory R:\DB2Profs.

The last **db2ilist** command shows that the instance has been moved to the cluster as we required.



```
DB2 CLP
C:\DB2\SQLLIB\BIN>
C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU
DB2CTLSV

C:\DB2\SQLLIB\BIN>dir R:
Volume in drive R is DB2 Disk 1
Volume Serial Number is 244D-7E5D

Directory of R:\

File Not Found

C:\DB2\SQLLIB\BIN>mkdir R:\DB2Profs

C:\DB2\SQLLIB\BIN>dir R:
Volume in drive R is DB2 Disk 1
Volume Serial Number is 244D-7E5D

Directory of R:\

11/03/2006 09:39 AM <DIR>          DB2Profs
                0 File(s)          0 bytes
                1 Dir(s)    53,602,504,704 bytes free

C:\DB2\SQLLIB\BIN>db2iclus MIGRATE /i:DB2PERU /p:R:\DB2Profs /c:ANCASH

DBI1912I The DB2 Cluster command was successful.

Explanation:
The user request was successfully processed.

User Response:
No action required.

C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU          C : ANCASH
DB2CTLSV
```

Figure 10-29 DB2 Instance migration

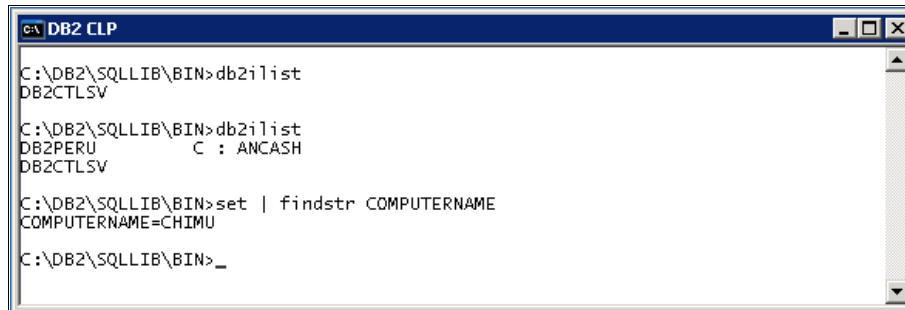
## 10.7.5 Adding a reference to the instance in the other nodes

The **db2iclus migrate** command places the DB2 instance profile directory in the shared drive and adds the instance to the shared part of the Windows registry. However, the instance was created in one node only (Mochica in this example). The instance creation process modifies several Windows registry keys as well as creating the service to be started in Windows. All these elements are missing from the other nodes in the cluster. If a failure occurs at this point, the Server Cluster will correctly failover to another node (Chimu) but Windows will not find the Windows Service for the DB2 instance DB2PERU. The failover process is aborted.

Use the `db2iclus` command with the `ADD` clause to create references to the instance in the remaining nodes. In our example, the second node is Chimu. The command must be run from the node that has the instance to be referenced and is as follows:

```
db2iclus ADD /i:DB2PERU /m:CHIMU /c:ANCASH /u:db2admin
```

Figure 10-30 shows that the `db2list` command is executed twice in machine Chimu. The second execution is after running the `db2iclus ADD` command showing that the instance is now existing in this node.



```
C:\DB2\SQLLIB\BIN>db2list
DB2CTLSV

C:\DB2\SQLLIB\BIN>db2list
DB2PERU      C : ANCASH
DB2CTLSV

C:\DB2\SQLLIB\BIN>set | findstr COMPUTERNAME
COMPUTERNAME=CHIMU

C:\DB2\SQLLIB\BIN>_
```

Figure 10-30 Adding a DB2 instance to the second node

Make sure that the correct ports are added in the `~system32/drivers/etc/services` file.

## 10.7.6 Creating the highly available DB2 instance resource

Migrating the instance to a cluster environment does not add this DB2 instance as a resource for the cluster. That is, up to this point, the cluster does not know that there is a highly available DB2 instance to be managed. In order to accomplish this, you must create a resource of type DB2 using Cluster Administrator utility.

Use the New Resource wizard in the Cluster Administrator to add the DB2 instance as a cluster resource:

1. From the Cluster Administrator, right-click the DB2 group created (DB2PERU-0 Group in this case) and then **File** → **New** → **Resource**.
2. The New Resource wizard shows up.

The resource name must be *exactly* the same as the last part of the Windows service name of the DB2 instance. In our example, the Windows service name of our instance is DB2-DB2PERU-0, so the resource name must be DB2PERU-0. Select **DB2** as the Resource Type and click **Next**. See Figure 10-31.

**Note:** If the resource name created is incorrect, the Server Cluster is not able to find the service to start or stop DB2. The resource becomes offline.

To change the resource name, you have to drop and recreate the resource with the correct name. Renaming the resource will not solve the problem.

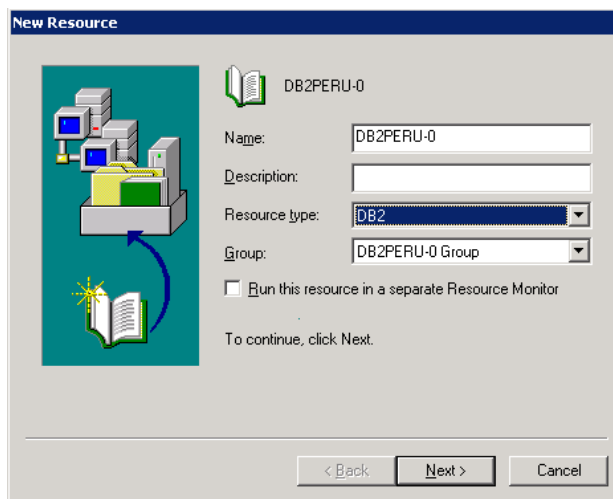


Figure 10-31 Creating the DB2 instance resource

3. In the Possible Owners window, accept the default of all nodes in the cluster as the owners. Click **Next**.
4. The Dependencies window is shown. The instance must be dependent in the highly available IP Address and disks created, as the instance cannot start without these two resources. They are DB2PERU-0 IP Address and Disk R:.

See Figure 10-32. Click **Next** to proceed. In the next dialog, click **Finish** to create the resource.

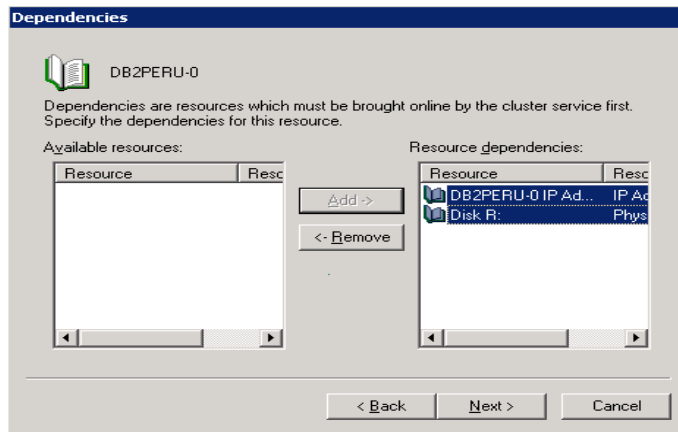


Figure 10-32 Dependency list for the DB2 instance resource

5. In the Cluster Administrator main window, right-click on the newly created resource and click **Bring online**.

After the DB2 instance is added to the cluster, you should start/stop the DB2 instance using only the Cluster Administrator interface. If you use any other means such as the DB2 Command Line Processor, you might have the following problems:

- ▶ If you stop the instance from the command line processor, the Server Cluster will detect this as a failure and will try to bring the instance online immediately.
- ▶ If the DB2 cluster service is offline and you bring the instance online from the command line processor, the cluster will not detect this and thus the instance will not be highly available.

## 10.8 Configuring a highly available DB2 instance using db2mscs

The **db2mscs** utility automates most of the steps of creating a highly available DB2 instance in the Windows Cluster environment. The prerequisite is that the shared disk where the instance profile is moved must be made highly available before running the utility. All other steps such as creating the highly available IP address and network name are automated by the tool.

**db2mscs** uses a configuration file that defines the user and password to create the services, the name and value of the resources like IP Addresses and network names, and so on. **db2mscs** accepts three parameters:

**db2mscs -f:<config\_file> -u:<instance name> -d:<debug\_file>**

- ▶ **-f** option: This option specifies the configuration file. The default value of this parameter is `db2mscs.cfg`
- ▶ **-u** option: This option allows you to revert all changes in the specified instance and return it to its unclustered form.
- ▶ **-d** option: This option specifies the debug file name. The utility writes debugging information into the text file specified in this option.

The configuration file accepts parameters in the following format:

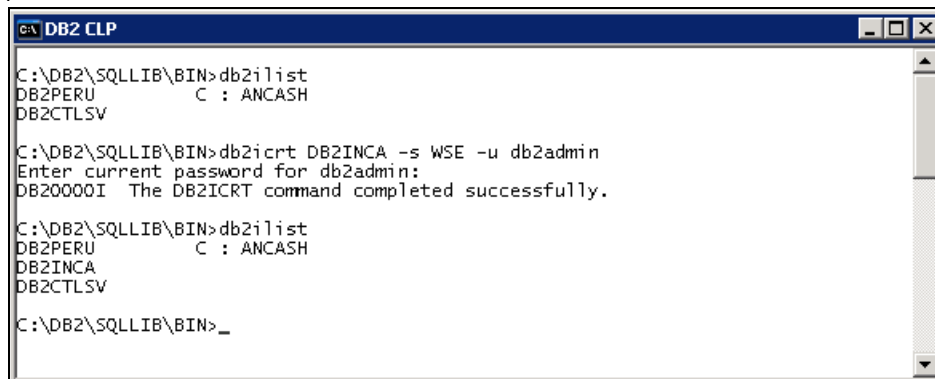
PARAMETER	VALUE
-----------	-------

DB2 provides some sample configuration files under the directory `%DB2Dir%\sqlib\cfg`:

- ▶ `db2mscs.das`, to cluster a DAS instance.
- ▶ `db2mscs.ee`, to cluster an EE instance.
- ▶ `db2mscs.eee`, to cluster an EEE instance.

The meaning of each parameter can be found in the DB2 manual, *Command Reference*, SC10-4226.

Because we have created an ESE instance in the previous example, we use `db2mscs` to cluster a WSE instance. Figure 10-33 shows the commands to a DB2 WSE instance named `DB2INCA`.



```
C:\DB2\SQLLIB\BIN>db2i list
DB2PERU      C : ANCASH
DB2CTLSV

C:\DB2\SQLLIB\BIN>db2icrt DB2INCA -s WSE -u db2admin
Enter current password for db2admin:
DB20000I The DB2ICRT command completed successfully.

C:\DB2\SQLLIB\BIN>db2i list
DB2PERU      C : ANCASH
DB2INCA
DB2CTLSV

C:\DB2\SQLLIB\BIN>_
```

Figure 10-33 Instance creation

The sample configuration file that we use is shown in Example 10-1.

*Example 10-1 Windows Cluster configuration file*

---

```
DB2_LOGON_USERNAME=CHAVIN\db2admin
DB2_LOGON_PASSWORD=mypasswd123
CLUSTER_NAME=ANCASH
GROUP_NAME=DB2INCA Group
DB2_INSTANCE=DB2INCA
#DB2_NODE=0

IP_NAME=DB2INCA IP Address
IP_ADDRESS=9.43.86.131
IP_SUBNET=255.255.255.0
IP_NETWORK=IBM Power 9

NETNAME_NAME=DB2INCA Net
NETNAME_VALUE=DB2INCA Net
NETNAME_DEPENDENCY=DB2INCA IP Address

DISK_NAME=Disk T:
INSTPROF_PATH=T:\DB2Profs
```

---

Here are some points to notice in this example:

- ▶ The value of IP\_NAME is arbitrary. It is a good practice to use the instance name for the IP\_NAME.
- ▶ The value of NETNAME\_DEPENDENCY has to be the same as the one in IP\_NAME because the network name depends on the IP\_NAME.
- ▶ The value of DISK\_NAME must be a physical disk resource that has already existed in the cluster.

Before running db2mscs, you *must* set the environment variable DB2INSTANCE to be the same value as the db2mscs.cfg parameter DB2\_INSTANCE. If you fail to do this, the operation will fail because db2mscs tries to start the sentence specified by DB2INSTANCE. Typically you will receive an error message such as:

```
C:\DB2\SQLLIB\BIN>db2mscs -f:\temp\db2mscs.cfg
DB21516E DB2MSCS cannot bring resource "DB2CTLSV" online. Ensure that
the properties of the resource are set correctly.
```

Changing the instance name will solve the problem, as we can see here:

```
C:\DB2\SQLLIB\BIN>set DB2INSTANCE=DB2INCA
C:\DB2\SQLLIB\BIN>db2mscs -f:\temp\db2mscs.cfg
```

```
DB21500I The DB2MSCS command completed successfully.
```



When these steps are completed, you can see the newly created resource in the Cluster Administrator as shown in Figure 10-34.

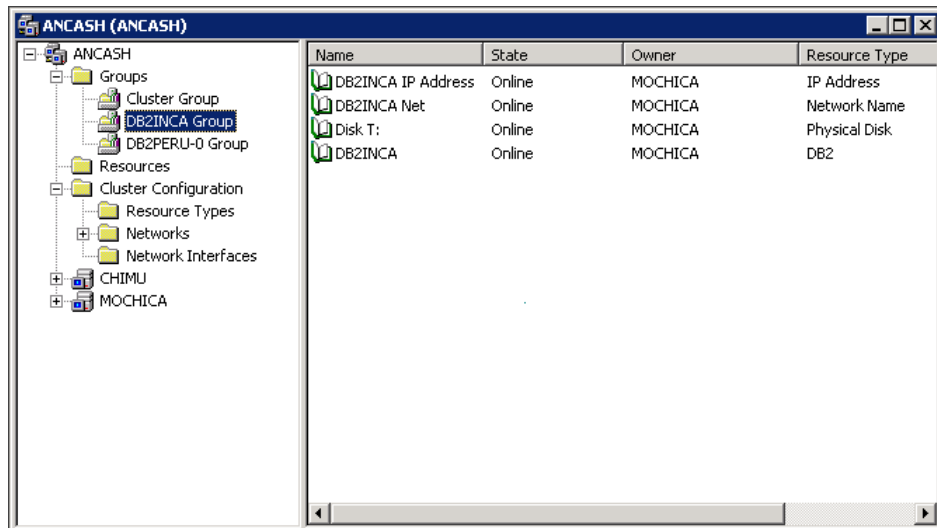


Figure 10-34 Final result of clustering a WSE instance





## HADR with clustering software

Clustering software such as High Availability Cluster Multi-Processing (HACMP) and Tivoli System Automation (TSA) are among system high availability options. Combining High Availability Disaster Recovery (HADR) with clustering software strengthens the high availability for the computing environment. In this chapter we discuss how to configure HADR with HACMP or TSA to enable automating HADR takeover.

We introduce the following topics:

- ▶ Why clustering software is needed
- ▶ Automating HADR takeover with HACMP
- ▶ Automating HADR takeover with TSA

## 11.1 Overview: Why clustering software is needed

The present version of HADR does not monitor the environment of the primary database server for outages such as network down. As illustrated in Figure 11-1, in the Remote Catchup Pending state, the standby database keeps waiting for log records to be transferred even though the primary database is no longer active. It is necessary to monitor the HADR pair and manually issue appropriate takeover commands in the event of a primary database server failure. This is where clustering software is required to automate HADR takeover.

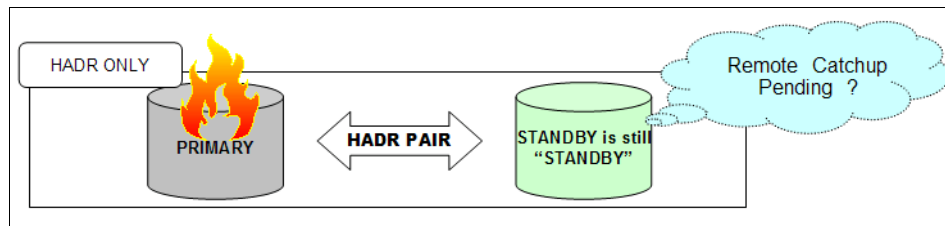


Figure 11-1 Takeover HADR will not happen automatically

### What is clustering software?

Clustering software automates failover of resources such as processes, applications, storages, and IP addresses. Clustering software monitors the health of the network, hardware and software processes, detects and communicates any fault, and automatically fails the service and associated resources over to a healthy host in the cluster.

In a cluster environment, the terms *failover*, *fallover*, and *takeover* are often used interchangeably, referring to the action whereby system activities are moved from the failed system to the healthy one. Specifically, failover refers to the activity of the broken node/server handing over responsibility to a backup or standby node/server. Fallover is similar, but a more general term for moving resources, including planned move operations for the purpose of maintenance. Takeover refers to the activity of the backup node/server making resources active after the original primary server breaks/fails.

There are several types of clustering software you can choose from, depending on the platforms where the databases are running. Here we list a few of them:

- ▶ Tivoli System Automation for Linux

Tivoli System Automation is a solution that is designed to provide a multi-tiered architecture of clustering in a heterogeneous environment. This can either be through the TSA base component on various platforms, or through existing base-level cluster solutions from other vendors,

coupled with the TSA end-to-end component, which interfaces with all the base-level clustering software (with adapters for non-TSA products). This enables central control of all clusters, and facilitates the grouping of interdependent base-level clusters into applications. TSA provides a consistent, single monitoring and control interface for these disparate servers and clusters within a mission-critical business environment, and can truly be considered an integrated and automated highly available solution.

For detailed information about Tivoli System Automation, see Chapter 8, “DB2 with TSA” on page 233.

- ▶ IBM High Availability Cluster Multi-Processing for AIX

HACMP offers robust high availability and disaster recovery for IBM customers with mission-critical applications. HACMP provides base services for cluster node membership, system management, configuration integrity and control, and failover and recovery for applications. HACMP clusters with both non-concurrent and concurrent access can be a System p, or a logical partition (LPAR) of an applicable System p.

For detailed information about HACMP, see the following Web site:

<http://ibm.com/systems/p/software/hacmp.html>

- ▶ Microsoft Cluster Service for Windows operating systems

Microsoft Cluster Service (MSCS) was introduced in Windows 2003. The new replacement for MSCS has improved capabilities: It allows you to connect up to eight nodes in a cluster; and, additional technology has been added to make possible geographically remote clusters.

For information about MSCS, see the following white paper, “Implementing IBM DB2 Universal Database™ V8.1 Enterprise Server Edition with Microsoft Cluster Server,” on the “DB2 Database for Linux, UNIX, and Windows Support” Web site:

<http://ibm.com/software/data/pubs/papers/>

- ▶ LifeKeeper® for Linux/Windows

LifeKeeper for Linux/Windows is a clustering software provided by SteelEye® Technology, Inc. LifeKeeper provides features and functions to monitor system and application health, maintaining client connectivity and providing uninterrupted data access. By maintaining the system uptime, LifeKeeper ensures the high availability for Linux/Windows applications.

More details of this product are available at:

<http://www.steeleye.com/products/>

The article, *DB2 UDB and SteelEye LifeKeeper for Linux - A High Availability Database*, introduces a sample configuration of integrating DB2 into LifeKeeper cluster environment. This article is available at the following Web site:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/lk.pdf>

► ClusterPerfect

DNCWARE ClusterPerfect is the clustering software of Toshiba Solutions Corporation. This product provides a way of keeping computing resources available to users in the event of a hardware or software failure. By connecting multiple servers, other servers in the cluster take over the workload when a server goes down.

### How HADR works in an environment with clustering software

When a primary database server outage occurs, clustering software detects the failure and fails the resources from the primary over to the standby node. You can configure the failover scripts to issue HADR takeover on the standby database in connection with the resource failover. Figure 11-2 shows a typical failover flow in the HADR cluster automated by the clustering software.

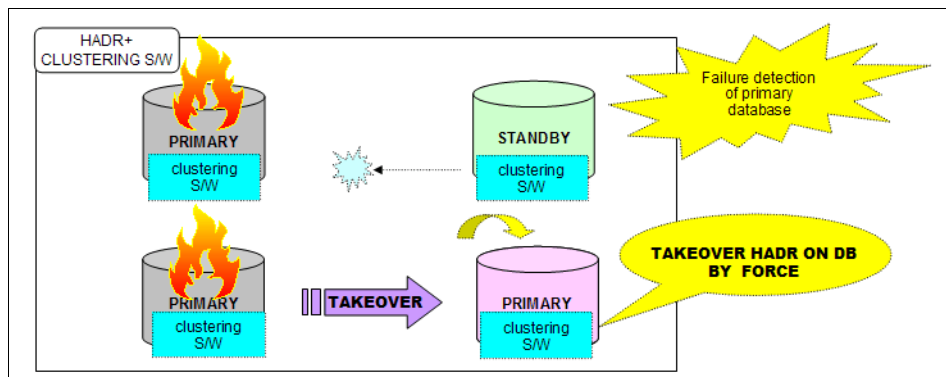


Figure 11-2 Why clustering software is needed to automate HADR takeover

In the cluster environment, the typical failover flow is as follows:

1. Under normal operations, users connect to the primary server and execute transactions against the database on this server. HADR continuously transfers the changes from the DB2 log from the primary to the standby server. In the standby, the logs are continuously replayed, thereby replicating the primary database. The standby database is in rollforward pending and no user can access this database.

2. In the event of an outage on the primary server, clustering software detects the fault, starts to failover the defined resources such as IP address, and executes takeover scripts. Clustering software can be configured to detect not only the server outage but also an instance crash by monitoring instance process.
3. The takeover script(s) issues an HADR takeover command against the standby database to change its role to primary. The old standby server can now serve users as the primary server.
4. While HADR takes care of synchronization of data between a primary and standby pair, there is still a need for applications to dynamically access the new primary server after an outage. This is achieved by the DB2 Automatic Client Reroute feature. Whenever the client fails to connect to the original server, the DB2 client will attempt to reroute the connection to the alternate server. If the reroute succeeds, the application can continue.
5. When the failed server comes back on line, it can then be started as the standby server, reversing roles to those prevailing prior to the outage.

### **What resources should be taken over**

The resources that must be taken over to automate HADR takeover with clustering software are as follows:

▶ **Storage devices:**

In a disk shared cluster, you must configure the shared storage device and create all database resources on it.

In an HADR environment, it is not necessary to share storage resources for failover, because the primary and standby databases are independent databases that reside on separate storage devices.

▶ **IP address:**

IP address takeover is optional. When you use Automatic Client Reroute, you do not always have to have a failover IP address because switching the IP address is handled on the client side. But be aware that for the clients or other servers which communicate with the database server and do not support Automatic Client Reroute, you must switch the definition of the IP address to the new primary database server.

▶ **Takeover scripts:**

Takeover scripts must be configured as a resource of the clustering software, and must be issued on the standby database by the clustering software at the time of takeover of resources after detecting an outage on the primary database.

## 11.2 Automating HADR takeover with HACMP

This section describes how to automate DB2 HADR takeover in an HACMP environment. The section also describes the failover test procedure for HADR controlled by the HACMP facility.

HACMP for AIX provides a highly available computing environment. HACMP facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. The primary reason to create HACMP clusters is to provide a highly available environment for mission-critical applications.

In an HACMP cluster, to ensure the availability of these applications, the applications are placed under HACMP control. HACMP ensures that the applications remain available even when a component in a cluster fails. To ensure availability in case of a component failure, the HACMP software moves the application along with the resources, to another node in the cluster. For more details regarding HACMP, see Chapter 8, “DB2 with TSA” on page 233.

### 11.2.1 HACMP and HADR planning

Before you set up an HADR and HACMP environment, you must plan the cluster environment. The following items must be considered:

- ▶ Physical nodes
- ▶ Network
- ▶ HACMP configuration
- ▶ HADR configuration
- ▶ Scripts



## Lab environment

Figure 11-3 shows the configuration of HACMP and HADR in our Lab environment.

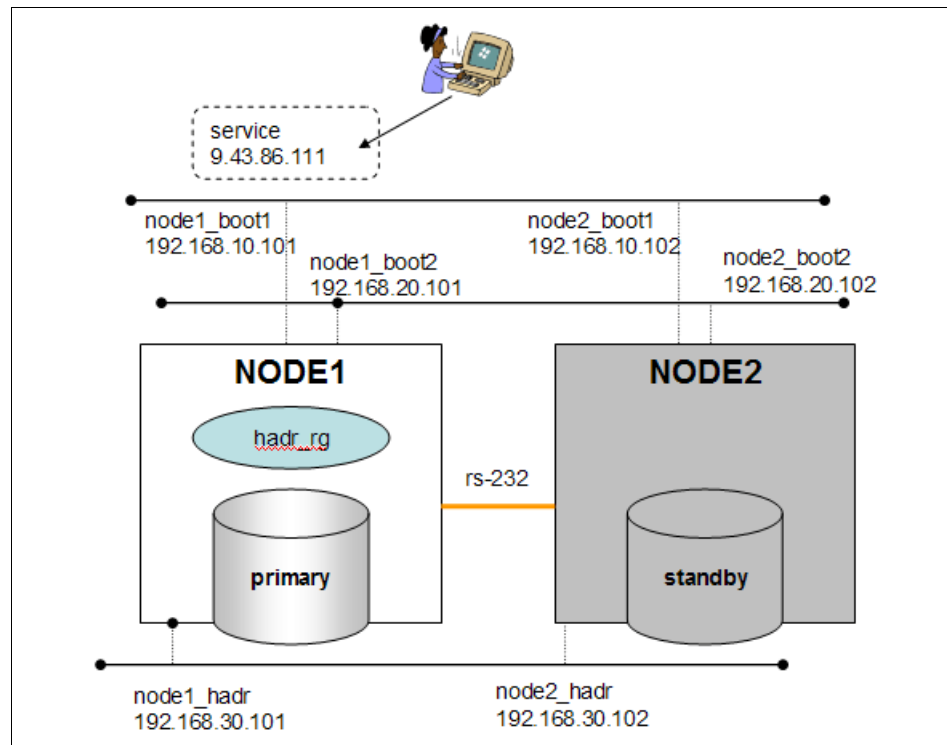


Figure 11-3 Lab environment

The planning for our Lab environment is as follows:

- ▶ Physical node configurations:
  - Two physical nodes named Node1 and Node2 are defined in the cluster.
  - Node1 is designated as the *service* node, which regularly provides service to clients. The HADR primary database runs on this node.
  - Node2 is the *standby* node on which the HADR standby database resides. The role of both these nodes changes in response to system failover or planned takeover issued by administrators.
- ▶ Network configurations:
  - Two ethernet network interfaces on each node are provided for the client's access, which are under the control of HACMP. The service address for the clients is added on one of these network interfaces.

- One ethernet network interface is dedicated to HADR communications. We recommend that you have a separate network for HADR if you want to avoid the interference of HADR log transfer.
- One serial (RS-232C) network is configured for HACMP keep-alive. Having a serial network (non TCP/IP network) is recommended in order to make HACMP failure detection more reliable.

**Note:** It is not necessary to have shared disks for HADR in an HACMP cluster, because the primary and standby databases are independent databases, which can reside in separate storage devices.

- ▶ HACMP configuration:
  - Resource group named *hadr\_rg* is configured, which includes a service address and an application server.
  - In our Lab, we configured a service address in the HACMP resource group. IP address takeover is optional if you use Automatic Client Reroute where clients can automatically switch the nodes they connect to.
- ▶ HADR configuration:
  - Each node has the DB2 instance named *hadrintst*. Instance names do not always have to be identical on both the nodes.
  - Each instance has the database named *SAMPLE*. The database name should be identical on both the nodes.
  - Configured dedicated network for HADR communication.

## Failover after primary node crash

Figure 11-4 shows the cluster behavior in the event of a primary (service) node crash:

1. HACMP detects the primary nodes outage and starts the failover process. The resource group *hadr\_rg* is acquired by the standby node (Node2).
2. The Service IP address related to the resource group is added to the standby node.
3. The Start script in the HACMP application server, which is related to the resource group, is issued on the standby node. The script includes the TAKEOVER HADR command to change the role of the standby database to primary.
4. Clients who address the service address succeed in connecting to the new primary database on the surviving node, which has the HACMP resource group now.

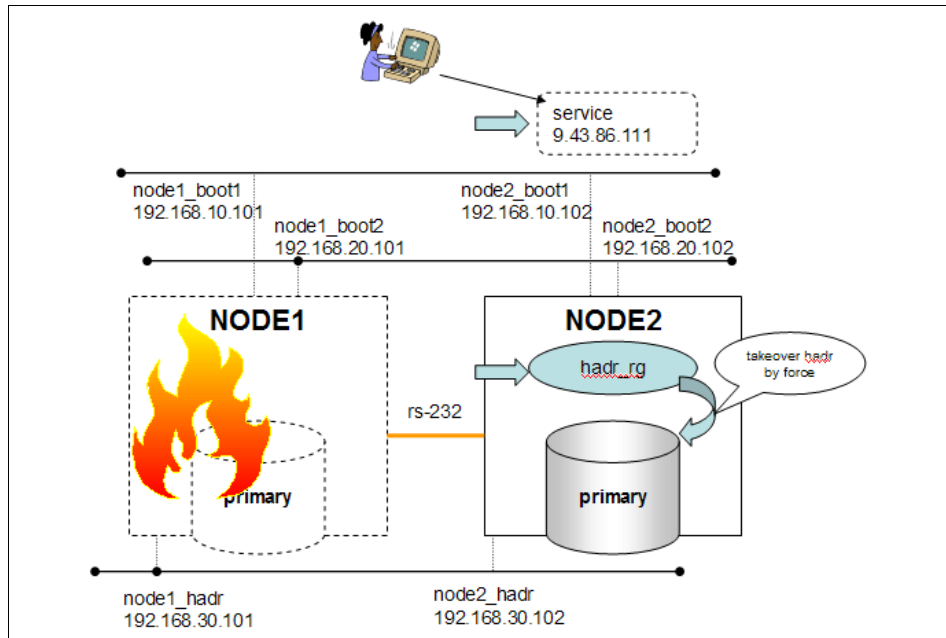


Figure 11-4 Automatic failover to standby node

## Reintegration of old primary node

Figure 11-5 shows the process required to reintegrate the old primary node into clusters. This process is as follows:

1. After Node1 is recovered from a hardware crash, you can start up the instance and start HADR as the standby database on this node by executing the **start hadr** command with the **as standby** option manually.
2. The standby database automatically catches up the log records which are processed only on the new primary database during the time the old primary Node is out of order.
3. After the standby database catches up all the log gaps, the HADR primary and standby again return to the Peer state. Reintegration of the old primary database hereby is complete.

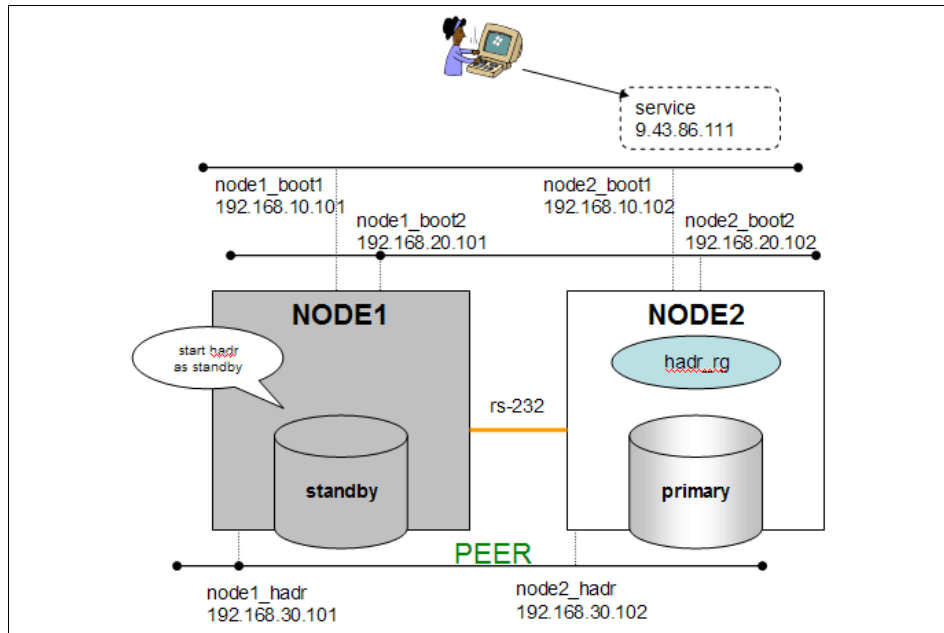


Figure 11-5 Reintegration of old primary database

## 11.2.2 Step-by-step configuration overview

You can configure automated HADR takeover environment with HACMP by completing the following operations:

1. Perform HADR setup.
2. Perform HACMP setup.
  - Here we test only the HACMP configuration and function with dummy scripts.
3. Prepare the scripts required to control HADR database from HACMP.
4. Perform a joint test of HADR and HACMP:
  - Normal start up of cluster
  - Planned takeover
  - Unplanned takeover
  - Normal stop of cluster

## 11.2.3 HADR setup

To configure the HADR database pair, complete the following steps:

1. For new systems, create a DB2 instance on both the nodes. We created `hadrinst`.
2. Check that proper entries are configured in `/etc/hosts` and `/etc/services` file.
3. For new systems, create a database on the primary node. We created `SAMPLE`.
4. Back up the primary database and restore the image on the standby node.
5. Configure HADR parameters properly on both the databases.
6. Start HADR on the standby database, and then start HADR on the primary database.
7. Check that both the databases can communicate with each other in the Peer state.

See Chapter 3, “HADR setup” on page 47 for more details on step-by-step configuration of HADR.

Figure 11-6 shows the entries for the services and hosts of the HADR configuration in our Lab environment.

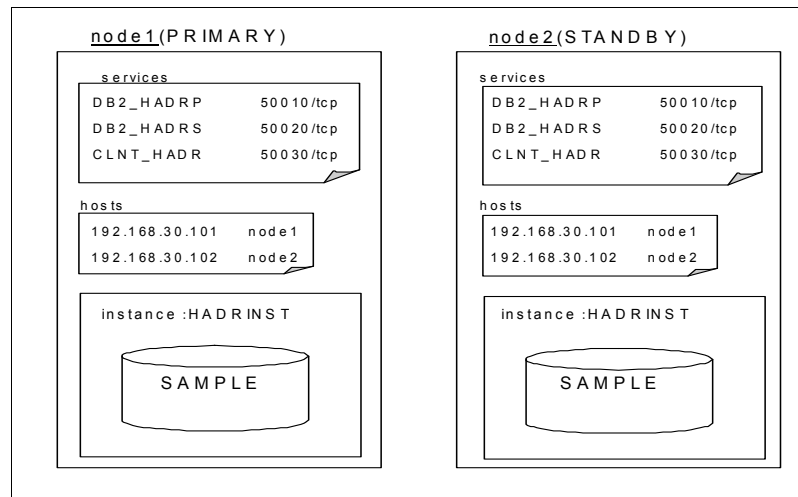


Figure 11-6 Instance, services, and hosts entries for HADR configuration

## 11.2.4 HACMP configuration

Here we describe the procedure to configure an HACMP cluster for automatic failure detection of the primary node.

### Checking TCP/IP network connection

HACMP relies on a TCP/IP network connection to communicate between nodes. Before you configure HACMP, check that the network is configured properly. To check network configurations for the cluster, complete the following steps:

1. Check whether the IP addresses are configured on network interfaces using the command:

```
#netstat -in | grep -v link
```

Example 11-1 shows the configuration of the IP addresses on both the nodes.

*Example 11-1 netstat output*

---

```
- node1 -
en0  192.168.10.101
en1  192.168.20.101

- node2 -
en0  192.168.10.102
en1  192.168.20.102
```

---

2. Check whether the `/etc/hosts` file has all the entries of IP addresses and that their labels are the same as used by HACMP:

```
#vi /etc/hosts
```

Example 11-2 shows an example of hosts file.

*Example 11-2 IP entries in hosts file*

---

```
9.43.86.111      service          ## Service IP address

192.168.10.101  node1_boot1     ## HACMP N/W interface on node1
192.168.20.101  node1_boot2     ## HACMP N/W interface on node1
192.168.30.101  node1_hadr      ## HADR N/W interface on node1

192.168.10.102  node2_boot1     ## HACMP N/W interface on node2
192.168.20.102  node2_boot2     ## HACMP N/W interface on node2
192.168.30.102  node2_hadr      ## HADR N/W interface on node2
```

---

3. Verify that name resolution is working well by using the **host** command. If something is wrong, check and modify the `/etc/hosts` file:

```
#host node1_boot
node1_boot1 is 192.168.10.101
```

4. Check the serial network connection:
  - a. Check that both machines are connected by RS232 cable.
  - b. Configure tty device on machine #1 and machine #2 using **smitty**:

```
# smitty tty
```

Select **Add a tty** → **tty rs232 Asynchronous Terminal** → **sa0 Available 01-S1 Standard I/O Serial Port**. In **Add a TTY** screen (Figure 11-7), press F4 to show the list of available port numbers. Select the port number displayed in the list.

```

                                     Add a TTY
[TOP]                                [Entry Fields]
TTY type                             tty
TTY interface                         rs232
Description                           Asynchronous Terminal
Parent adapter                         sa0
* PORT number                          [0] +
Enable LOGIN                           disable +
BAUD rate                               [] +
PARITY                                  [none] +
BITS per character                       [8] +
Number of STOP BITS                     [1] +
TIME before advancing to next port setting [0] +#
TERMINAL type                            [dumb]
FLOW CONTROL to be used                  [xon] +
OPEN DISCIPLINE to be used               [dtropen] +
[MORE...30]
```

Figure 11-7 Add a TTY

- c. Check that the tty device is available as shown here:

```
# lsdev -Cc tty
tty0 Available 01-S1-00-00 Asynchronous Terminal
```

## Configuring HACMP

AIX `smitty` provides HACMP configuration interfaces. The main steps for this are as follows:

1. Add nodes to an HACMP cluster.
2. Add a service IP label/address.
3. Configure application servers.
4. Add a resource group.
5. Add resources to the resource group.
6. Configure persistent IP alias for HADR communication.
7. Define the serial network and the serial network device.
8. Verify and synchronize HACMP configurations.
9. Conduct a basic verification of HACMP configuration.

Next we provide the details of the steps to be followed:

1. Add nodes to an HACMP cluster:

```
#smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Add Nodes to an HACMP Cluster**

In Configure Nodes to an HACMP Cluster (standard) menu (Figure 11-8), enter the cluster name and new nodes.

```
Configure Nodes to an HACMP Cluster (standard)

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                     [Entry Fields]
* Cluster Name                       [hadr_cluster]
New Nodes (via selected communication paths) [node1_boot1 node2_boot1] +
Currently Configured Node(s)
```

Figure 11-8 Add node to HACMP cluster



2. Add a service address to the resource group:

```
# smitty hacmp
```

From HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Service IP Labels/Addresses** → **Add a Service IP Label/Address**.

In the Add a Service IP Label/Address (standard) menu, enter the IP label and the network name as shown in Figure 11-9.

```

                                     Add a Service IP Label/Address (standard)
Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                     [Entry Fields]
* IP Label/Address                    [service ]                +
* Network Name                        [net_ether_01]              +
```

Figure 11-9 Add a service IP label

3. Prepare the application server start/stop scripts.

In this example, we use dummy start and stop scripts just for testing the HACMP setup. We provide the full functional start and stop scripts to control HADR database in an HACMP environment in the next section.

Example 11-3 is a dummy start/stop script which just records the time when the scripts are executed. We place the scripts in /home/hadrinst/scripts directory. Copy this dummy script and change its name as follows:

- Start script of HACMP application server:  
/home/hadrinst/scripts/hadr\_primary\_takeover.ksh
- Stop script of HACMP application  
server:/home/hadrinst/scripts/hadr\_primary\_stop.ksh

*Example 11-3 Dummy application server start/stop script*

---

```
#!/usr/bin/ksh -x

exec >> /tmp/~basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

exit 0
```

---

Make sure that these scripts have execution permissions:

```
#ls -l /home/hadrinst/scripts
```

4. Configure the application server.

To access this menu, proceed as follows:

```
# smitty hacmp
```

From HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Application Servers** → **Add an Application Server**.

In the Add Application Server menu, enter the server name, and complete the paths of the start and stop scripts, as shown in Figure 11-10.

Add Application Server

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

* Server Name	[hadr_server ]
* Start Script	[/home/hadrinst/scripts/hadr_primary_takeover.ksh ]
* Stop Script	[/home/hadrinst/scripts/hadr_primary_stop.ksh ]

*Figure 11-10 Adding application server*

5. Add a resource group.

To add a resource group, go to **smitty**:

```
# smitty hacmp
```

From HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Add a Resource Group**.

In the Add a Resource Group menu, enter the Resource group name and the participating node names, as shown in Figure 11-11. This corresponds to a *Rotating resource group* in the earlier HACMP versions, which means that there is no priority for a resource group between nodes. For further details, see *HACMP for AIX 5L V5.3 Planning and Installation Guide*, SC23-4861-08, Chapter 6.

Add a Resource Group

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

[Entry Fields]

* Resource Group Name	[ <i>hadr_rg</i> ]	
* Participating Node Names (Default Node Priority)	[ <i>node1 node2</i> ]	+
Startup Policy	<i>Online Using Distribution Policy</i>	+
Fallover Policy	<i>Fallover To Next Priority Node in The List</i>	+
Fallback Policy	<i>Never Fallback</i>	+

Figure 11-11 Adding a resource group

6. Add resources to the resource group:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Change/Show Resources for a Resource Group (standard)**.

In the Change/Show Resources for a Cascading Resource Group menu, enter service IP label/address, service, as shown in Figure 11-12.

```

Change/Show Resources for a Cascading Resource Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                     [Entry Fields]
Resource Group Name                   hdr_rg
Participating Node Names (Default Node Priority)  node1 node2

Startup Policy                         Online Using Distribution Policy
Failover Policy                        Fallover To Next Priority Node In The List
Fallback Policy                         Never Fallback

Service IP Labels/Addresses            [service] +
Application Servers                     [hdr_server]
+
Volume Groups                           []
+
Use forced varyon of volume groups, if necessary  false +
Filesystems (empty is ALL for VGs specified)    [] +

```

Figure 11-12 Adding resource to resource group

7. Define serial N/W and serial network device.

The subsidiary network for HACMP keep-alive is recommended to make HACMP failure detection more secure. For more details, see *HACMP 5.4 Administration Guide*, SC23-4862-09. Enter:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Extended Configuration** → **Extended Topology Configuration** → **Configure HACMP Communication Interfaces/Devices** → **Add Communication Interfaces/Devices** → **Add Discovered Communication Interface and Devices** → **Communication Devices**.

In the Select Point-to-Point Pair of Discovered Communication Devices to Add menu, choose the tty device on each node you connected with RS-232C cable. In our example, we select node1 tty2 and node2 tty2.

8. Verify and synchronize HACMP configurations.

After defining the HACMP configuration from one node, you must verify and synchronize the cluster topology to the other node.

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Verify and Synchronize HACMP Configuration**.

The HACMP verification utility checks that the cluster definitions are the same on all the nodes and provides diagnostic messages if errors are found.

## Starting HACMP

HACMP configuration is now complete. The next step is to start up the HACMP cluster as follows:

1. Start HACMP on service node Node1.

First you start up HACMP on service node Node1. To start up HACMP from the **smi t** menu, complete these steps:

```
# smitty clstart
```

In the Start Cluster Services menu, select **now** on Start now, on system restart or both; select **true** on Startup Cluster Information Daemon?; and leave the default for all the other fields, as shown in Figure 11-13.

```
Start Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                     [Entry Fields]
* Start now, on system restart or both      now      +
Start Cluster Services on these nodes      [node1 ]   +
BROADCAST message at startup?              false     +
Startup Cluster Information Daemon?         true      +
Reacquire resources after forced down ?    false     +
```

Figure 11-13 Starting cluster services

Check cluster.log to see if HACMP is started successfully by using the following command:

```
# tail -f /usr/es/adm/cluster.log
```

The message **EVENT COMPLETED: node\_up\_complete xxx** indicates that HACMP is started successfully, where xxx is the node number. Figure 11-14 shows the output of cluster.log.

```

Oct 17 20:04:00 node1 HACMP for AIX: EVENT START: node_up node1
Oct 17 20:04:02 node1 HACMP for AIX: EVENT START: acquire_service_addr
Oct 17 20:04:05 node1 HACMP for AIX: EVENT START: acquire_aconn_service en0 net_ether_01
Oct 17 20:04:05 node1 HACMP for AIX: EVENT COMPLETED: acquire_aconn_service en0 net_ether_01 0
Oct 17 20:04:05 node1 HACMP for AIX: EVENT COMPLETED: acquire_service_addr 0
Oct 17 20:04:06 node1 HACMP for AIX: EVENT COMPLETED: node_up node1 0
Oct 17 20:04:08 node1 HACMP for AIX: EVENT START: node_up_complete node1
Oct 17 20:04:09 node1 HACMP for AIX: EVENT START: start_server hadr_server
Oct 17 20:04:10 node1 HACMP for AIX: EVENT COMPLETED: start_server hadr_server 0
Oct 17 20:04:10 node1 HACMP for AIX: EVENT COMPLETED: node_up_complete node1 0

```

Figure 11-14 Cluster.log output

2. Start HACMP on standby node Node2.

After the HACMP service node is started, you can use the same procedure to start the standby node.

After HACMP is started on both the nodes, check the following items:

- ▶ Check the status of the resource group using the HACMP command: **c1RGinfo**. You can issue this command on either node. Example 11-4 shows the out put of this command. You can see that resource group hadr\_rg is OFFLINE on node1 and ONLI NE on node2, which means that the resource group hadr\_rg is now acquired by node2.

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
```

Example 11-4 Check resource group status

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
-----
Group Name      State           Node
-----
hadr_rg         OFFLINE        node1
                 ONLINE         node2
-----
```

- ▶ Check if the service address is added on service node Node1.

```
#netstat -i | grep -v link
```

Example 11-5 shows the output of the **netstat** command. The service IP should be in the output list.

*Example 11-5 Check service address*

---

```
root@node1:/home/hadrinst/scripts# netstat -i | grep -v link
Name Mtu Network Address Ipkts Ierrs Opkts Oerrs
Coll
en0 1500 192.168.10 node1_boot1 3268881 0 2956143 0
0
en0 1500 9.43.86 service 3268881 0 2956143 0
0
en1 1500 192.168.20 node1_boot2 793555 0 1671197 0
0
```

---

- ▶ On the service node, make sure that the application server is started.

When the HACMP is started, the application server Start script is executed. Check the log written by the Start script. In our example, it is `hadr_primary_takeover.ksh.log`.

```
# cat /tmp/hadr_primary_takeover.ksh.log
```

Example 11-6 shows the sample output. Because now we have just set a dummy script, nothing happens on HADR databases. The script only records the time when the script is issued to the log file.

*Example 11-6 Check application server*

---

```
hadr_rg:[6] echo
#####
#####
hadr_rg:[7] date
Tue Oct 17 20:04:10 CDT 2006
```

---

## HACMP takeover test

You can test the HACMP takeover function by stopping HACMP on the service node in takeover mode. The standby node should take over the resource group after the takeover operation. To stop HACMP in the takeover mode, use **smitty** in service node, Node1 in our example:

```
#smitty clstop
```

In the Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node where HACMP is to be stopped; and select **takeover** on Shutdown mode. See Figure 11-16.

```

Stop Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                     [Entry Fields]
* Stop now, on system restart or both      now                +
  Stop Cluster Services on these nodes     [node1 ]              +
  BROADCAST cluster shutdown?             false                 +
* Shutdown mode                            takeover               +

```

Figure 11-15 Stop HACMP with takeover

- ▶ Check whether the HACMP takeover process has stopped the service node. The HACMP takeover process writes a message to cluster.log. The node\_down\_complete xxx event message in the log indicates that the takeover event is now complete in the service node, Node1 as shown in Example 11-7.
 

```
# tail -f /usr/es/adm/cluster.log
```

Example 11-7 Check takeover result

```

# tail -f /usr/es/adm/cluster.log
.....
Oct 20 20:59:55 node1 HACMP for AIX: EVENT COMPLETED:
node_down_complete node1
.....

```

- ▶ Check that the application server is stopped on service node Node1. View the application server stop script log to see whether the application server has been stopped successfully:
 

```
# cat /tmp/hadr_primary_stop.ksh.log
```
- ▶ On the service node Node1, check if the service address is released:
 

```
# netstat -i
```
- ▶ On standby node Node2, check that node\_down\_complete Node1 event is completed:
 

```
# tail -f /usr/es/adm/cluster.log
```



Example 11-8 shows a snippet of the cluster.log.

*Example 11-8 Check node\_down\_complete event from Node 2*

---

```
# tail -f /usr/es/adm/cluster.log
.....
Oct 20 20:59:55 node2 HACMP for AIX: EVENT COMPLETED:
node_down_complete node1
.....
```

---

- ▶ Check that Node2 owns the resource group using the `clRGinfo` command as shown in Example 11-9.

*Example 11-9 Checking if Node2 has taken over the resource*

---

```
# /usr/es/sbin/cluster/utilities/clRGinfo
```

Group Name	State	Node
hadr_rg	OFFLINE	node1
	ONLINE	node2

---

- ▶ Check that Node2 has taken over the service IP address. Issue the `netstat` command in Node2 as shown in Example 11-10.

*Example 11-10 Check service IP address*

---

```
root@node2:/home/hadrinst/scripts# netstat -i | grep -v link
```

Name	Mtu	Network	Address	Ipkts	Ierrs	Opkts	Oerrs
Co11							
en0	1500	192.168.10	node2_boot1	3268881	0	2956143	0
0							
en0	1500	9.43.86	service	3268881	0	2956143	0
0							
en1	1500	192.168.20	node2_boot2	793555	0	1671197	0
0							

---

- ▶ Check that the application server is started on Node2.  
Because our start/stop scripts are still dummy, HADR takeover is not issued yet. Here we just checked if HACMP functions work correctly:

```
# cat /tmp/hadr_primary_takeover.ksh.log
```

## Stopping HACMP

To stop HACMP, use `smitty` on both Node1 and Node 2.

```
#smitty clstop
```

In the Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node to be stopped; and select **graceful** for Shutdown mode, as shown in Figure 11-16.

```

Stop Cluster Services

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[Entry Fields]

* Stop now, on system restart or both      now      +
Stop Cluster Services on these nodes      [node1]  +
BROADCAST cluster shutdown?              false    +
* Shutdown mode                           graceful  +

```

Figure 11-16 Stop HACMP

## 11.2.5 Prepare application server scripts

To automate HADR takeover with HACMP, we utilize the HACMP application server start/stop scripts. By including the HADR commands in the start/stop scripts, HACMP can handle HADR operations in connection with the resource group. The application scripts for HADR can be placed in the directory of your choice. We place our scripts in the `/home/hadrinst/scripts` directory. The sample scripts can be found in Appendix A, “HACMP application server scripts” on page 809. Here we explain some of these HACMP controlled scripts and how they work.

### ***hadr\_primary\_takeover.ksh (start script)***

We define a shell script `hadr_primary_takeover.ksh` as the start script for HACMP application server. This script is executed in the following situations when the node acquires the resource group:

- ▶ Normal start up of HACMP on the first (service) node in the cluster.
- ▶ Unplanned takeover that is triggered by failure detection of HACMP.
- ▶ Planned takeover by stopping HACMP in takeover mode, or moving the resource group by an administrative command.

This script is executed not only when the resource group has fallen over to the standby node, but also when the first node in the cluster is started and acquires the resource group. This script checks which is the starting trigger by issuing the HACMP command: `c1RGinfo`.

Figure 11-17 shows the flow chart of the script, *hadr\_primary\_takeover.ksh*. If the trigger is normal HACMP start up, then this script simply exits. If the starting trigger is takeover, which means that this script is issued on the standby node in connection with the resource group fall over, then this script checks the HADR role. If the HADR role is standby, it issues the **takeover hadr** command with *by force* option.

In circumstances where the standby database cannot communicate with the primary database, the **takeover hadr** command needs the *by force* option to change its role to primary. But for planned takeover, *by force* option is not preferable, so in our example, the normal **takeover hadr** command (without the *by force* option) is issued from the stop script by a remote command before this script is issued on the standby node.

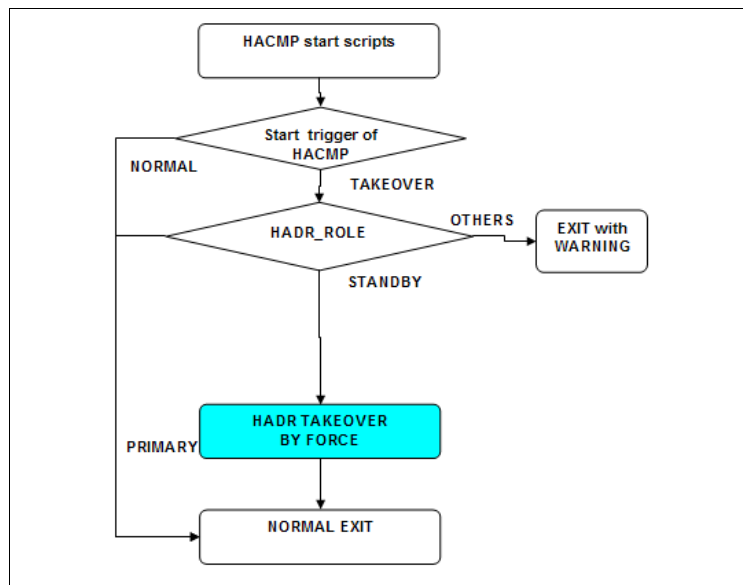


Figure 11-17 *hadr\_primary\_takeover.ksh* script flow

### ***hadr\_primary\_stop.ksh* (stop script)**

We define *hadr\_primary\_stop.ksh* as the stop script for HACMP application server. Figure 11-18 shows the script logic flow. This script is executed on the service node and when the node releases the resource group. That is, when you stop HACMP in takeover mode or move the resource group from one server to the other intentionally for planned takeover. For planned takeover, it is preferable to issue the **takeover hadr** command as soon as possible. In our example, this script issues a remote command to the standby node to execute the **takeover hadr** command.

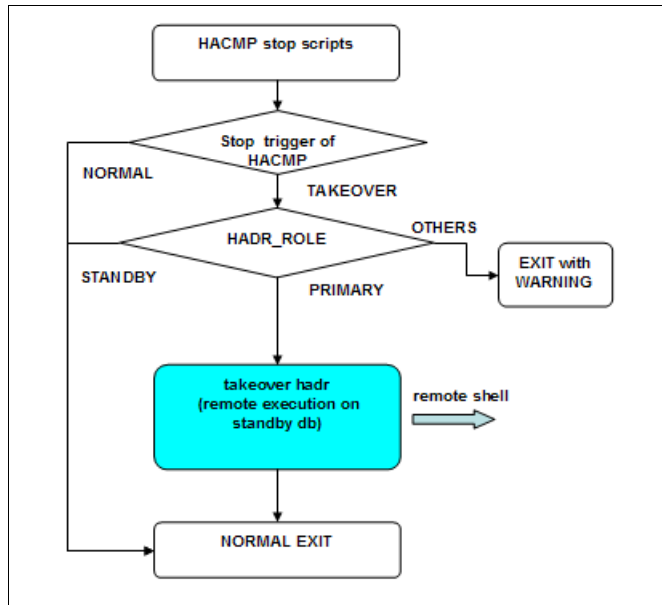


Figure 11-18 *hadr\_primary\_stop.ksh* script flow

### **Additional information for HACMP commands**

You can use the output of the HACMP command `c1RGinfo` to define the start trigger of the application server:

```
/usr/es/sbin/cluster/utilities/c1RGinfo -a
```

In the Resource Group Movement column, you can see just one node when the start trigger is normal startup (Example 11-11); two nodes when the start trigger is takeover (Example 11-12).

*Example 11-11 Normal start up on node1*

```
# /usr/es/sbin/cluster/utilities/c1RGinfo -a
-----
-----
Group Name      Type           Resource Group Movement
-----
-----
hadr_rg         non-concurrent PRIMARY=" :node1"
```

*Example 11-12 Takeover from node1 to node2*

```
# /usr/es/sbin/cluster/utilities/clRGinfo -a
-----
Group Name      Type           Resource Group Movement
-----
hdr_rg          non-concurrent PRIMARY="node1:node2"
```

## 11.2.6 Joint test for HADR and HACMP

After the functional scripts have replaced the dummy scripts, we can test the combination of HADR automatic takeover in an HACMP environment. We implement the joint test for the following scenarios:

► **Planned failover:**

In the first scenario, we use the facility provided by the HACMP application to test the failover between the two nodes.

► **Unplanned failover:**

For the unplanned failover scenario, we halt the primary system to simulate a real system outage and verify that the failover occurs properly.

### Starting up HADR and HACMP in the cluster

To carry out the test scenarios, complete the following steps to start up HADR and HACMP:

1. Start standby database on node2.

Start up the HADR databases you have already set up in 11.2.3, “HADR setup” on page 355. Start the standby database on Node2 first:

- Check that HADR role is standby on node2:

```
$db2 get db cfg for sample
```

- Start up the standby database:

```
$db2 activate db sample
```

**Tip:** When you start up the HADR process, you do not have to issue **start hadr** or **stop hadr** command every time. After you start an HADR database in a primary or standby role, just activate and deactivate database can be used to start and stop the HADR database.

2. Start the primary database on Node1:
  - Check that HADR role is primary on node1:  
`$db2 get db cfg for sample`
  - Start up the standby database:  
`$db2 activate db sample`
  - Check if HADR status is in the Peer state:  
`$db2pd -hadr -db sample`
3. Start HACMP on Node1.  
 For details, See “Starting HACMP” on page 363.
4. Start HACMP on Node2.  
 For details, See “Starting HACMP” on page 363.
5. Check HADR and HACMP status and they are ready for the test:
  - Check if HADR is in the Peer state:  
`$db2pd -hadr -db sample`
  - Check if resource group is ONLINE on node1:  
`# /usr/es/sbin/cluster/utilities/c1RGinfo`

### Configuring the client node

To check the client access after takeover, configure the client machine by completing the steps listed below. We configured the client machine on Node3.

1. Create instance on client node:  
`$db2icrt -u clntinst clntinst`
2. Catalog node directory specifying the service address: service  
`$db2 catalog tcpip node SERVICE remote service server 50030`
3. Catalog database:  
`$ db2 catalog database sample at node SERVICE`
4. Connect to database server:  
`$ db2 connect to sample user hadrinst using hadrinst`
5. Type the query shown in Example 11-13 to check which server you are connecting to. You can see that we are now connected to Node1.

*Example 11-13 Check the server connected*

---

```
$ db2 "select substr(host_name,1,8) as hostname from table
(db_partitions()) as t"
```

```
HOSTNAME
```

```
-----
```

```
node1
```

```
1 record(s) selected.
```

---

## **Planned failover and fallback**

In the first scenario, we use the facility provided by the HACMP application to test the failover between the two nodes:

1. Stop HACMP in takeover mode on service node, Node1:

```
#smitty clstop
```

In Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node where HACMP is to be stopped; select **takeover** on Shutdown mode.

2. Check if HACMP takeover process has stopped the service node.

The HACMP takeover process writes a message to cluster.log. The node\_down\_complete xxx event message in the log indicates that the takeover event is completed in the service node, Node1.

```
# tail -f /usr/es/adm/cluster.log
```

3. Check that the **stop** script is issued on Node1 and the TAKEOVER HADR command was issued remotely on Node2. You will see the output as in Example 11-14 in the log file of this script.

```
#cat hadr_primary_stop.ksh.log
```

*Example 11-14 Output of stop script*

---

```
....
```

```
HADR takeover is issued on remote node
```

```
.....
```

```
node2: DB2000I The TAKEOVER HADR ON DATABASE command completed
successfully.
```

---

4. Check the current status of HACMP on Node2. The resource group should be taken over to Node2. See Example 11-15.

*Example 11-15 Check who owns resource group*

---

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
```

```
-----  
Group Name      State           Node  
-----  
hadr_rg         OFFLINE        node1  
                ONLINE         node2  
-----
```

5. Check the status of the HADR database on both the nodes. Now the primary database is running on Node2 and the standby database is running on Node1:

```
$db2pd -hadr -db sample
```

6. Reconnect from the client node, Node3, and issue the query as shown in Example 11-16. You can see that you have connected to node2. In this case, you must reconnect to the server after you receive a communication error. If the alternate server is set, reconnection is automatically done by Automatic Client Reroute.

*Example 11-16 Check reconnected node*

---

```
$ db2 "select substr(host_name,1,8) as hostname from table  
(db_partitions()) as t"
```

```
HOSTNAME
```

```
-----
```

```
node2
```

```
1 record(s) selected.
```

---

7. Start HACMP on Node1 again. Check that there is no change in the HADR status and the resource group is still ONLINE on Node2.
8. Fallback from Node2 to Node1:  
To fallback the resource group from Node2 to Node1, you can stop HACMP in takeover mode on Node2.
9. Start HACMP on Node2: The primary database is running on Node1 and the standby database is running on Node2. Now the cluster has returned to the state we started with.

## Unplanned failover for system crash

In this scenario, we test the failover by halting the system intentionally. We check that HACMP detects the failure and that the standby database is switched to the primary database in connection with the HACMP resource group takeover.



1. Check the HACMP and HADR status of the two nodes:
  - HACMP service is started on both nodes.
  - HADR is in the Peer state.
2. Connect from the client and issue some queries following the steps shown in “Configuring the client node” on page 372.
3. On Node1, issue the following command as root:
 

```
#sync;sync;sync;halt -q
```

Node1 will terminate processing immediately. Soon HACMP detects the outage and starts failover of the resource group to Node2.
4. Check that the resource group is taken over by Node2:
 

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
```
5. Check that the start script is issued on Node2.
 

```
# view /tmp/hadr_primary_takeover.ksh.log
```
6. Check that the primary HADR database is now on Node2
 

```
$ db2pd -hadr -db sample
```

If the HADR primary database is on Node2, the failover is successful.

Next, we want to failback the resource group to the original node, as the failed system situation has been repaired. Power on Node1 and start DB2 instance.

```
$db2start
```

You have to reintegrate the old primary database to the current state using the HADR function. In the old primary system (Node1), do the following actions:

1. Check the HADR role on Node1: The HADR role on Node1 should still show as primary. See Example 11-17.

*Example 11-17 Check HADR role*

---

```
$ db2 get db cfg for sample | grep "HADR database role"
```

```
HADR database role = PRIMARY
```

---

2. Check the db2diag.log file using the following command to see the HADR catching up process. After you start the HADR database as standby on Node1, the catching up process will start.

```
$ db2diag -f
```

or

```
$ tail -f /home/hadrinst/sql1lib/db2dump/db2diag.log
```

3. Restart the HADR database as standby on Node1.

To reintegrate the database on Node1 into HADR pair as a standby database, complete the following steps:

- a. To switch the role of the database from primary to standby, you have to issue the **start hadr** command instead of **activate database**. See Example 11-18. Check the messages in the db2diag.log until they return to the peer status.

*Example 11-18 Start database as HADR standby*

---

**\$db2 "start hadr on db sample as standby"**

DB20000I The START HADR ON DATABASE command completed successfully.

---

- b. Check whether the HADR status has come back to the Peer state after catching up the log from the primary on Node1 as follows:
  - The **db2pd** command shows the current HADR status.
  - In the db2diag.log file you will see messages as shown in Example 11-19.

*Example 11-19 db2diag.log message*

---

```
2006-10-30-23.49.47.617235+540 E353590C344          LEVEL: Event
PID      : 43462                TID   : 1          PROC  : db2hadrs
(SAMPLE) 0
INSTANCE: hadrinst             NODE   : 000        DB    : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE   : HADR state set to S-Peer (was S-NearlyPeer)
```

---

4. Start HACMP on Node1 again. Check that there is no change in the HADR status and the resource group is still ONLINE on Node2.
5. Fallback from Node2 to Node1: To fallback the resource group from Node2 to Node1, you can stop the HACMP in takeover mode on Node2.
6. Start HACMP on Node2.

After the primary database is running on Node1 and the standby database is running on Node2, you can start HACMP on Node2. The cluster now returns to the state we started from.

## Stopping HACMP and HADR

Complete these steps to stop the HACMP system for a maintenance operation:

1. Stop HACMP on the Service node and the standby node.

See “Stopping HACMP” on page 367.

## 2. Stop HADR:

- Stop HADR on the primary database on the service node:  
`$db2 deactivate db sample`
- Stop instance on the service node:  
`$db2stop`
- Stop HADR on the standby database on the standby node:  
`$db2 deactivate db sample`
- Stop instance on the standby node:  
`$db2stop`

**Note:** The **deactivate database** command can be used to stop the HADR process instead of the **stop hadr** command. When you issue **deactivate database**, the HADR database role stays in the database configuration. If you issue the **stop hadr** command, the HADR database role changes to STANDARD, which makes it hard for you to know which node was primary and which node was standby.

## 11.3 Automating HADR takeover with TSA on AIX

In this section we describe how to configure automatic DB2 HADR takeover with TSA on an AIX environment.

TSA ensures that the applications remain available even when a component in a cluster fails. To ensure availability in case of a component failure, TSA moves the application along with the resources, to another node in the cluster. In an HADR with TSA environment, TSA facilitates the detection of failures and automatically executes the HADR takeover from one system to another in the cluster after a hardware or software failure. In a TSA cluster, HADR is placed under TSA control. For more details regarding TSA, see Chapter 8, “DB2 with TSA” on page 233.

In DB2 V8.2 and V9.1, clustering software is not automatically integrated with the DB2 product. Instead, users must manually combine the two with appropriate scripts for the clustering software to correctly monitor and manage DB2 HADR resources. In this section, we describe the manual configuration of HADR with TSA with DB2 9.1, where TSA is not supplied with the DB2 package, and does not provide the integrated High Availability Feature released in DB2 V9.5.

The configuration of DB2 9.5 HADR with TSA is described in Chapter 12, “Configuring clusters using the DB2 9.5 High Availability Feature” on page 477. With the DB2 9.5 HA configuration tool **db2haicu**, all necessary resources, dependencies, and equivalencies are automatically defined to TSA during the configuration process.

### 11.3.1 Architecture

Before you set up an HADR with TSA environment, you must plan the cluster environment. Consider the following items:

- ▶ Physical nodes:
  - Decide the primary and the secondary nodes. The primary node (or service node) regularly provides service to clients. The HADR primary database runs on this node. The role of both nodes can be changed in response to system failover or planned takeover issued by administrators.
  - Confirm software requirements on each node.
- ▶ Network:
  - Network interface that provides client access on each node. This is one of the TSA controlled resources.
  - Have one network card for TSA tiebreaker. We recommend that you have a separate network for HADR to avoid the interference of HADR log transfer.

**Note:** It is not necessary to have shared disks for HADR in an TSA cluster, because the primary and standby databases are independent databases, which can reside in separate storage devices.

- ▶ TSA configuration:

Define the cluster domain name that includes the resources of virtual IP (service IP), DB2 instance, and HADR database.
- ▶ HADR configuration:

Define the HADR configuration. Refer to Chapter 5, “Automatic client reroute” on page 121 to define an HADR configuration.
- ▶ Scripts:

To automate HADR takeover with TSA, we utilize the TSA start, monitor, and stop scripts. To consider the scripts, refer to “Preparing scripts for TSA” on page 394.

## Lab environment

Figure 11-19 shows the configuration of HADR with TSA in our lab environment.

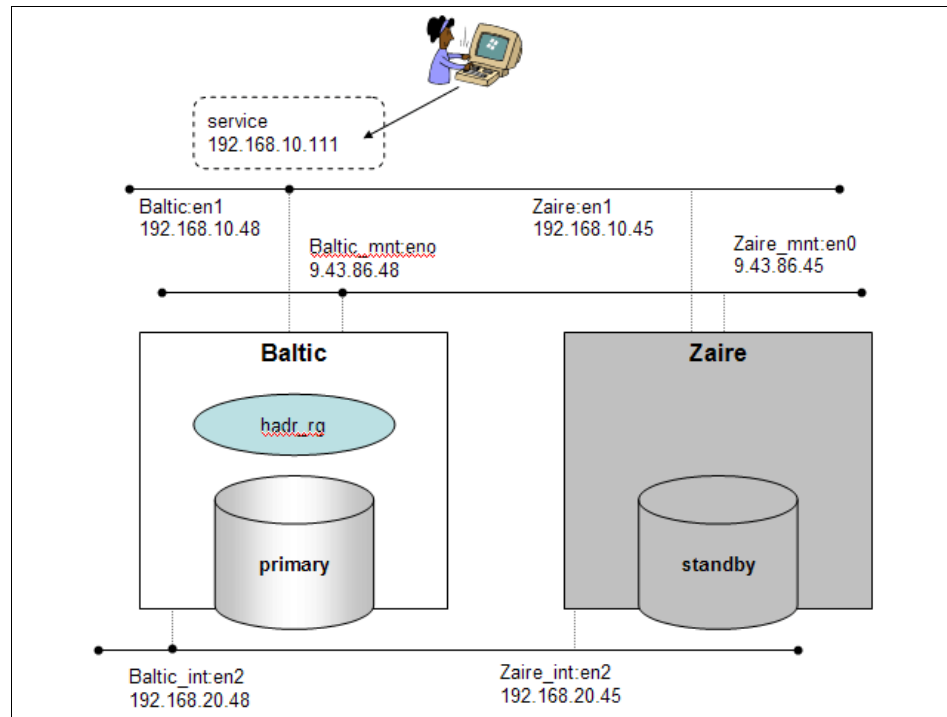


Figure 11-19 HADR with TSA lab environment

The planning for our Lab environment is as follows:

- ▶ Physical node configurations:
  - Two physical nodes named Baltic and Zaire are defined in the cluster.
  - Baltic is designated as the *service* node, which regularly provides service to clients. The HADR primary database runs on this node.
  - Zaire is the *standby* node on which the HADR standby database resides. The role of both these nodes changes in response to system failover or planned takeover issued by administrators.
  - The following software configuration is used to set up the lab environment:
    - AIX 5.3 Technology Level 7
    - DB2 9.1
    - Tivoli System Automation for Multiplatforms (SA MP) Base Component V2.2

- ▶ Network configurations:
  - One ethernet network interface (en1) on each node is provided for the client’s access, which is under the control of TSA. The service address for the clients is added on one of these network interfaces:
    - Primary node (Baltic)
    - en1: 192.168.10.48 (255.255.252.0)
    - Standby node (Zaire)
    - en1: 192.168.10.45 (255.255.252.0)
  - One ethernet network interface (en2) is dedicated to HADR communications:
    - Primary node (Baltic)
    - en2: 192.168.20.48 (255.255.252.0)
    - Standby node (Zaire)
    - en2: 192.168.20.45 (255.255.252.0)
  - One ethernet network is configured for the TSA tiebreaker. We use en1 for the tiebreaker. Refer to 8.1.2, “Terminology of TSA” on page 236 for a detailed description of a tiebreaker of TSA.
- ▶ TSA configuration:
  - A resource group named *hadr\_rg* is configured, which includes a service address and an application server.
  - In our Lab, we configured a service address in the TSA resource group. IP address takeover is optional if you use Automatic Client Reroute where clients can automatically switch the nodes they connect to.
- ▶ HADR configuration:
  - Each node has the DB2 instance named *db2inst1*.
  - Each instance has the database named *SAMPLE*. The database name should be identical on both the nodes.
  - Configured dedicated network for HADR communication.
- ▶ Scripts
 

TSA uses scripts to start, stop, and monitor resources. In this section, we create the customized shell scripts. Refer to “Preparing scripts for TSA” on page 394.

## 11.3.2 Configuration

You can configure the automated HADR takeover environment with TSA by completing the following operations:

1. Install DB2.
2. Install TSA.
3. Perform HADR setup.
4. Perform TSA setup.
  - Here we test only the TSA configuration and function with dummy scripts.
5. Prepare the scripts required to control HADR database from TSA.
6. Perform a joint test of HADR with TSA:
  - Normal start up of cluster
  - Planned takeover
  - Unplanned takeover
  - Normal stop of cluster

In our examples, we use the notations (P) and (S) preceding a command to designate on which node that command is to be issued to properly set up the topology as shown in Figure 11-19 on page 379. The order of the notation also designates the sequence of the command execution on the nodes.

- ▶ (P): Primary database node (Baltic)
- ▶ (S): Standby database node (Zaire)
- ▶ (P)(S): Command issued on the primary node first, then the secondary node

### H/W and S/W prerequisites

The minimum software requirement for running DB2 on AIX can be found at the following Web site:

<http://ibm.com/software/data/db2/9/sysreqs.html>

For information about software requirements for running TSA, refer to:

<http://ibm.com/software/tivoli/products/sys-auto-linux/platforms.html>

- ▶ Tivoli System Automation V2.2 manuals can be found at the Web site:  
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>
- ▶ RSCT for AIX manuals can be found at the Web site:  
[http://publib.boulder.ibm.com/infocenter/clresctr/vrxr/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct\\_aix5153/b15adm1110.html](http://publib.boulder.ibm.com/infocenter/clresctr/vrxr/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct_aix5153/b15adm1110.html)

## The other prerequisites

The time and dates on the standby and the primary nodes must be synchronized as closely as possible. This is absolutely critical to ensure smooth failover for the primary node failures.

## HADR setup

To configure the HADR database pair, complete the following steps:

1. For new systems, create a DB2 instance on both the nodes. We created `db2inst1`.
2. Check that proper entries are configured in `/etc/hosts` and `/etc/services` file.
  - Make sure that the host names (Baltic, Zaire) and the HADR internal network names (Baltic\_int, Zaire\_int) are included in `/etc/hosts`.
  - Make sure that the HADR communication ports (`hadrp_db2inst1`, `hadrs_db2inst1`) and the DB2 client/server communication ports (`db2c_db2inst1`) are included in `/etc/services`.
3. For new systems, create a database on the primary node. We created `SAMPLE`.
4. Back up the primary database and restore the image on the standby node.
5. Configure HADR parameters properly on both databases.
6. Start HADR on the standby database, and then start HADR on the primary database.
7. Check that both databases can communicate with each other in the Peer state.

See Chapter 3, “HADR setup” on page 47 for detailed step-by-step HADR configuration.

Figure 11-20 shows the entries for the services and hosts of the HADR configuration in our Lab environment.



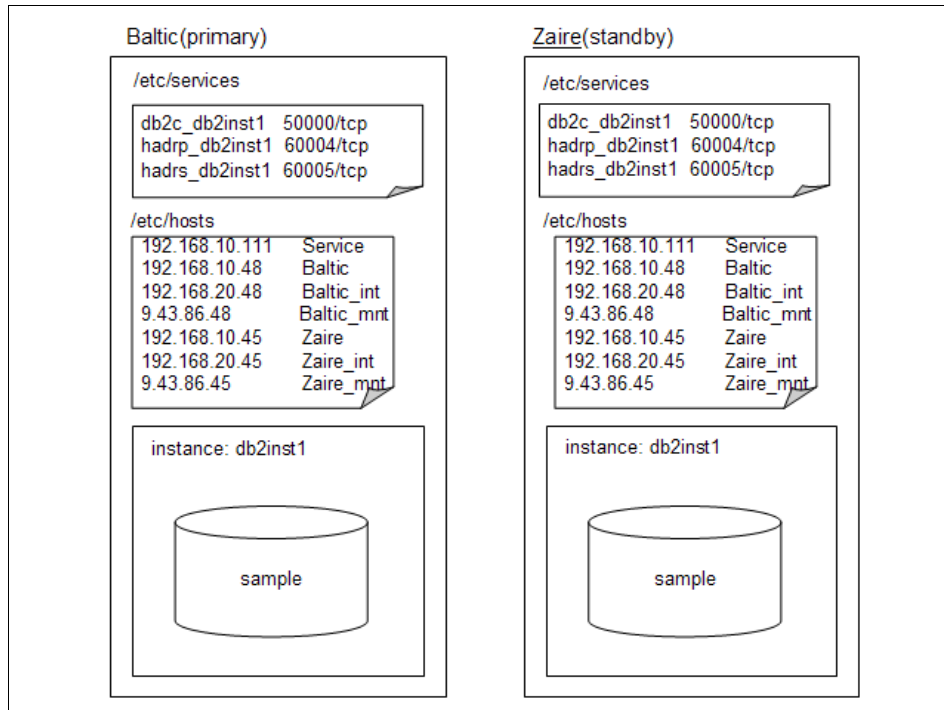


Figure 11-20 Configuration of HADR

## Checking the TCP/IP network connection

Before you configure TSA, check that the network is configured properly. To check network configurations for the cluster, complete the following steps:

1. Check whether the IP addresses are configured on the network interfaces using the command:

```
netstat -in | grep -v link
```

Example 11-20 shows the configuration of the IP addresses on Baltic.

*Example 11-20 IP address configurations*

---

```
(P)(S) #netstat -in | grep -v link
Name Mtu Network Address
en0 1500 9.43.84 9.43.86.48
en1 1500 192.168.8 192.168.10.48
en2 1500 192.168.20 192.168.20.48
lo0 16896 127 127.0.0.1
```

---

2. Check whether the `/etc/hosts` file has all the entries of IP addresses.

Example 11-21 shows the content of the `hosts` file in `Baltic`. `Zaire` should have the same entries.

*Example 11-21 Sample of hosts file*

---

```
(P)(S)#vi /etc/hosts

192.168.10.111  Service    ## Virtual IP address

192.168.10.48  Baltic     ## TSA N/W interface for Virtual IP on
Baltic
192.168.20.48  Baltic_int ## HADR N/W interface on Baltic
9.43.86.48     Baltic_mnt ## For Maintenance N/W interface on
Baltic

192.168.10.45  Zaire      ## TSA N/W interface for Virtual IP on
Zaire
192.168.20.45  Zaire_int  ## HADR N/W interface on Zaire
9.43.86.45     Zaire_mnt  ## For Maintenance N/W interface on
Zaire
```

---

3. Verify that name resolution is working well by using the `host` command. If something is wrong, check and modify the `/etc/hosts` file:

```
(P)(S)#host Baltic
Baltic is 192.168.10.48
```

## SSH setup

Many of the TSA commands that you issue in the configuration steps require `ssh` to be set up on all nodes. In this topic we describe how to set up `ssh` for the root user so that no password is required when commands are entered. For additional information, consult the following article:

<http://ibm.com/developerworks/db2/library/techarticle/dm-0506finnie/>

Log on to the root and perform the following procedures on all nodes:

1. If the `/.ssh` directory does not yet exist, create it. Ensure that `/.ssh` directory does *not* allow write access for group or other.
2. From the `/.ssh` directory, generate a public key/private key pair.

```
(P)(S) #ssh-keygen -t rsa
```

When prompted for input, press Enter. Do not enter a passphrase.

3. To enable the new key pair for use with `ssh`, execute these commands:

```
(P)(S) #mv /.ssh/id_rsa /.ssh/identity
(P)(S) #chmod 600 /.ssh/identity
(P)(S) #cat /.ssh/id_rsa.pub >> /.ssh/authorized_keys
(P)(S) #chmod 644 /.ssh/authorized_keys
(P)(S) #rm /.ssh/id_rsa.pub
```

4. Use the `ssh-keyscan` utility to gather the public host key for each host. Substitute the IP addresses in the following sample commands with your IP addresses used:

```
(P)(S) #ssh-keyscan -t rsa Baltic_mnt,9.43.86.48,
Baltic,192.168.10.48, Baltic_int,192.168.20.48 >> /.ssh/known_hosts
(P)(S) #ssh-keyscan -t rsa Zaire_mnt,9.43.86.45,
Zaire,192.168.10.45, Zaire_int,192.168.20.45 >> /.ssh/known_hosts
```

5. Test that the passwordless `ssh` is now enabled. From all nodes, execute the following command:

```
(P)(S) #ssh Zaire date
```

Make sure that this command succeeds without any prompting asking for additional verification.

## Setup for the cluster domain of TSA

The main steps for setting up the cluster domain are as follows:

1. Do initial configurations.
2. Prepare shell scripts.
3. Configure cluster domain.
4. Verify the operational quorum and tiebreaker definition.
5. Configure resource of IP address.
6. Configure resource of HADR.
7. Configure equivalency resource.
8. Configure resource group.
9. Configure relationship of resource.
10. Start resource group.

### *Initial configurations*

On all nodes, set the AIX environment variable `CT_MANAGEMENT_SCOPE` to 2 (peer domain scope). All TSA users can permanently set `CT_MANAGEMENT_SCOPE=2` by adding the following statement in the `./profile` file:

```
export CT_MANAGEMENT_SCOPE=2
```

### *Prepare shell scripts*

You can use the sample start, stop, and monitor scripts provided by DB2 or TSA. If you use the sample script provided by DB2, refer to 11.4, “Automating HADR

takeover with TSA on Linux” on page 405. In this scenario, we use the customized scripts on TSA. The details are described in “Preparing scripts for TSA” on page 394. Note that they are “as-is” and are non-supported scripts.

The scripts for HADR can be placed in the directory of your choice. We place our scripts in the `/usr/sbin/rsct/sapolicies/hadr/` directory. All of them should be set as executable.

### ***Configure the cluster domain***

Make sure that all TSA nodes in your topology know about one another, and can communicate with one another in what is referred to as a TSA cluster domain.

Perform these steps to configure the cluster domain:

1. Run the **preprnode** command as root to prepare the local node for joining the domain on each of the nodes:

```
preprnode nodename1 nodename2
```

```
(P)(S) # preprnode Baltic Zaire
```

2. Issue the following command to create the cluster domain from either the primary or the standby node:

```
mkrpdomain domainname nodename1 nodename2
```

```
(P) # mkrpdomain hadr_domain Baltic Zaire
```

3. Now start the cluster domain from either the primary or the standby node as follows:

```
starttrpdomain domainname
```

```
(P) # starttrpdomain hadr_domain
```

Note that all future TSA commands to be run are relative to this active domain.

4. After a few seconds, the `hadr_domain` is online. Use **lsrpdomain** to check that `hadr_domain` status:

```
(P)(S) # lsrpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
hadr_domain Online  2.4.8.3           No           12347 12348
```

5. Use **lsrpnode** to check that all nodes in the domain are online:

```
(P)(S) # lsrpnode
Name OpState RSCTVersion
baltic Online  2.4.8.3
zaire Online  2.4.8.3
```

6. Modify the default setting of the failure detection time:

The default setting of the failure detection time is usually short. It can cause misjudgment of failure detection (false positives), resulting in unnecessary frequent failover. We change the setting of the detection time.

We can view the current settings with the **lscmg** command, which lists information about the communication groups in a peer domain.

Communication groups control how liveliness checks (in other words, topology services' heartbeats) are performed between the communication resources within the peer domain.

```
(P)(S)# lscmg
```

Name	Sensitivity	Period	Priority	Broadcast	SourceRouting
CG1	4	1	1	Yes	Yes
CG2	4	1	1	Yes	Yes
CG3	4	1	1	Yes	Yes

```
(P)(S) #lscmg -i CG1
```

Name	NodeName	IPAddress	Subnet	SubnetMask
en0	Zaire	9.43.86.45	9.43.84.0	255.255.252.0
en0	Baltic	9.43.86.48	9.43.84.0	255.255.252.0

A communication group's *Sensitivity* setting refers to the number of missed topology services heartbeats that constitute a failure. A communication group's *Period* setting refers to the number of seconds between topology services' heartbeats.

The current Sensitivity is four times and the current Period is one second. These values mean that we begin takeover 8 ( $=4*1*2$ ) seconds after the failure occurs. It is short, so we have to modify the settings.

To modify the sensitivity setting of a communication group, use the **chcomg** command with the **-s** flag on one node. To modify the period setting of a communication group, use the **chcomg** command with **-p** flag:

```
(P) #chcomg -s 8 CG1
```

```
(P) #chcomg -s 8 CG2
```

```
(P) #chcomg -s 8 CG3
```

```
(P) #chcomg -p 2 CG1
```

```
(P) #chcomg -p 2 CG2
```

```
(P) #chcomg -p 2 CG3
```

```
(P)(S)# lscmg
```

Name	Sensitivity	Period	Priority	Broadcast	SourceRouting
CG1	8	2	1	Yes	Yes
CG2	8	2	1	Yes	Yes
CG3	8	2	1	Yes	Yes

### **Verify the operational quorum and tiebreaker definition**

In order to protect data, the configuration manager uses a quorum of nodes (called an operational quorum) to determine whether resources can be safely activated without creating conflicts with other subsystems.

In the case of a tie in which the peer domain has been partitioned into two sub-domains containing exactly half of the defined nodes, the configuration resource manager uses a tie-breaker resource (instance of the IBM.TieBreaker resource class) to determine which sub-domain has an operational quorum.

In this scenario, we configure the network tiebreaker by the `mkrsrc` command on one node. Network tiebreaker support is described in further detail in *IBM Tivoli System Automation for Multiplatforms Base Component User's Guide*, SC33-8272-00. Here is an example:

```
(P) #mkrsrc IBM.TieBreaker Type="EXEC" Name="nettb"  
DeviceInfo="PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=192.168.10.51  
Log=1"
```

```
(P) #lsrsrc -Ab IBM.TieBreaker  
Resource Persistent and Dynamic Attributes for IBM.TieBreaker  
resource 1:  
    Name                = "nettb"  
    Type                 = "EXEC"  
    DeviceInfo           = "PATHNAME=/usr/sbin/rsct/bin/samtb_net  
Address=192.168.1.51 Log=1"  
    ReprobeData          = ""  
    ReleaseRetryPeriod   = 0  
    HeartbeatPeriod      = 0  
    PreReserveWaitTime   = 0  
    PostReserveWaitTime  = 0  
    NodeInfo             = {}  
    ActivePeerDomain     = "hadr_domain"  
    ConfigChanged        = 0
```

To make `nettb` the active tiebreaker, use the `chrsrc` command with its `-c` flag:

```
chrsrc -c IBM.PeerNode OpQuorumTieBreaker=TieBreakername
```

To list the current active tiebreaker, use the `lsrsrc` command with `-c` flag:

```
lsrsrc -c IBM.PeerNode OpQuorumTieBreaker
```

```
(P) #chrsrc -c IBM.PeerNode OpQuorumTieBreaker=nettb
```

```
(P) #lsrsrc -c IBM.PeerNode  
Resource Class Persistent Attributes for IBM.PeerNode
```

```

resource 1:
    CommittedRSCTVersion = ""
    ActiveVersionChanging = 0
    OpQuorumOverride = 0
    CritRsrcProtMethod = 1
    OpQuorumTieBreaker = "nettb"
    QuorumType = 0
    QuorumGroupName = ""
    Fanout = 32

```

In addition to the tiebreaker definition, we require some additional configuration changes so that TSA can detect network interface failures. On every node in the domain, we create the following file: `/usr/sbin/cluster/netmon.cf`. (in TSA V2.3, the directory is changed to `/var/cf/cfg/netmon.cf`).

Each line of the `netmon.cf` file should contain a host name or IP address. TSA automatically uses the IP address or host name that exists in the same subunit as the appropriate network interface. We recommend that you use the IP address of the gateway for that network interface:

```

(P)(S)#vi /usr/sbin/cluster/netmon.cf
192.168.10.51

```

### ***Configure resource of IP address***

To configure the resource of virtual IP (Service IP) address, we use the `mkrsrc` command:

```

mkrsrc IBM.ServiceIP Name=ipname
NodeNameList="{nodename1,nodename2}" IPAddress=ipaddress
NetMask=subnetmask

```

Where:

- ▶ `ipname`: Any name for Virtual IP address
- ▶ `ipaddress`: Virtual IP address
- ▶ `subnetmask`: Sub netmask address

```

(P) #mkrsrc IBM.ServiceIP Name="ip1"
NodeNameList="{Baltic,Zaire}" IPAddress=192.168.10.111
NetMask=255.255.252.0

```

### ***Configure resource of HADR***

We use the `mkrsrc` command to create the resource of HADR. All resource characteristics can be provided in command line parameters as well as a definition file in plain text. We use a definition file named `hadr.def` as shown in Example 11-22.

```
PersistentResourceAttributes::  
Name="db2_hadr"  
NodeNameList="{ 'Baltic', 'Zaire' }"  
StartCommand="/usr/sbin/rsct/sapolicies/hadr/hadr_start.ksh"  
StopCommand="/usr/sbin/rsct/sapolicies/hadr/hadr_stop.ksh"  
MonitorCommand="/usr/sbin/rsct/sapolicies/hadr/hadr_monitor.ksh"  
MonitorCommandPeriod=30  
MonitorCommandTimeout=30  
StartCommandTimeout=300  
StopCommandTimeout=300  
UserName="root"
```

---

The resource definition can now be created with the `mkrsrc` command using the definition file.

```
(P) #mkrsrc -f hadr.def IBM.Application
```

```
(P) #lsrsrc -Ab IBM.Application  
Resource Persistent and Dynamic Attributes for IBM.Application  
resource 1:  
Name="db2_hadr"  
ResourceType= 1  
AggregateResource="0x3fff 0xffff 0x00000000..."  
StartCommand= /usr/sbin/rsct/sapolicies/hadr/hadr_start.ksh"  
StopCommand= "/usr/sbin/rsct/sapolicies/hadr/hadr_stop.ksh"  
MonitorCommand ="/usr/sbin/rsct/sapolicies/hadr/hadr_monitor.ksh"  
MonitorCommandPeriod= 30  
MonitorCommandTimeout= 30  
StartCommandTimeout= 300  
StopCommandTimeout= 300  
UserName= "root"  
NodeNameList= {"Baltic", "Zaire"}  
OpState=2
```

### **Configure equivalency resource**

To configure the equivalency resource, we use the `mkequ` command.

```
mkequ -p 0 equivalencyname  
IBM.NetworkInterface:interfacename1:hostname1,interfacename2:hostname2
```

The following command creates a so-called static equivalency named `equ` which contains a network adapter from each node of the cluster.

```
(P) # mkequ -p 0 equ IBM.NetworkInterface:en1:Baltic,en1:Zaire  
(P) # lsequ -Ab
```



## Configure resource group

By adding a resource into a resource group, this resource is automated and controlled by TSA. The following examples show how the resources `db2_hadr` and `ip1` are added into the same resource group.

To make the resource group, use the `mkrng` command.

```
mkrng rname
(P) # mkrng hadr_rg
```

To add the resources into the resource group, use the `addrngmbr` command.

```
addrngmbr -g rname resourceclass:resourcename ...
```

Both resources `db2_hadr` and `ip1` are added to the resource group `hadr_rg`. Adding the resources to the resource group makes them the managed resources:

```
(P) #addrngmbr -g hadr_rg IBM.Application:db2_hadr IBM.ServiceIP:ip1
```

To confirm the resource group, we use the `lsrg` command. The resources are added in the resource group as Example 11-23.

### Example 11-23 Output of `lsrg`

---

```
(P) #lsrg -m
Displaying Member Resource information:
Class:Resource:Node[ManagedResource] Mandatory MemberOf OpState
WinSource Location
IBM.Application:db2_hadr          True      hadr_rg  Offline
IBM.ServiceIP:ip1                True      hadr_rg  Offline
```

---

## Configure relationship of resource

There are two conditions that relate resources `db2_hadr` and `ip1` to one another. First, both resources must be started and be available on the same node in the cluster. Furthermore, it is of no use to start the DB2 HADR `db2_hadr` on a node, until the IP address `ip1` is established. TSA provides a relationship type called `DependsOn` that gathers both required conditions.

A managed relationship is defined with the `mkrel` command:

```
mkrel -p DependsOn -S IBM.Application:resource1 -G
IBM.ServiceIP:resource2 relationshipname
```

Example 11-24 creates a managed relationship named `hadr_dependson_ip1` that establishes the dependency of resource `db2_hadr` on the IP address `ip1` and a managed relationship named `ip1_dependson_equ` that establishes the dependency of resource `ip1` on the `equ`.

*Example 11-24 the mkrel command*

---

```
(P) #mkrel -p DependsOn -S IBM.Application:db2_hadr -G
IBM.ServiceIP:ip1 db2_hadr_dependson_ip1
(P) #mkrel -p DependsOn -S IBM.ServiceIP:ip1 -G IBM.Equivalency:equ
ip1_dependson_equ
```

---

### **Start resource group**

To start the resource group, use the **chrg** command. When the resource group is started, it is ready for service. See Example 11-25. The setup is completed.

```
chrg -o online resourcegroupname
```

*Example 11-25 chrg command and output of lssam*

---

```
(P) # lssam
Offline IBM.ResourceGroup:hadr_rg Nominal=Offline
    |- Offline IBM.Application:db2_hadr
        |- Offline IBM.Application:db2_hadr:Baltic
        '- Offline IBM.Application:db2_hadr:Zaire
    '- Offline IBM.ServiceIP:ip1
        |- Offline IBM.ServiceIP:ip1:Baltic
        '- Offline IBM.ServiceIP:ip1:Zaire

(P) # chrg -o online hadr_rg
(P) # lssam
Online IBM.ResourceGroup:hadr_rg Nominal=Offline
    |- Online IBM.Application:db2_hadr
        |- Online IBM.Application:db2_hadr:Baltic
        '- Offline IBM.Application:db2_hadr:Zaire
    '- Online IBM.ServiceIP:ip1
        |- Online IBM.ServiceIP:ip1:Baltic
        '- Offline IBM.ServiceIP:ip1:Zaire
```

---

Here we show the following meanings of the states in the **lssam** output:

- ▶ For instance resources (db2\_hadr):
  - Online: Indicates instance is up at the nodename.
  - Offline: Indicates instance is down at the nodename.
- ▶ For virtual IP resources (ip1):
  - Online: Indicates virtual IP is up at the node name.
  - Offline: Indicates virtual IP is down at the node name.

### **Resource groups topology**

The relationships between resources are illustrated in Figure 11-21. The resources illustrated correspond to the resources shown in the **lssam** output:

- ▶ HADR database resource group: *hadr\_rg* (1ssam)
  - Member Resources:
    - HADR DB resource: *db2\_hadr* (1ssam)
    - Virtual IP address resource: *ip1* (1ssam)

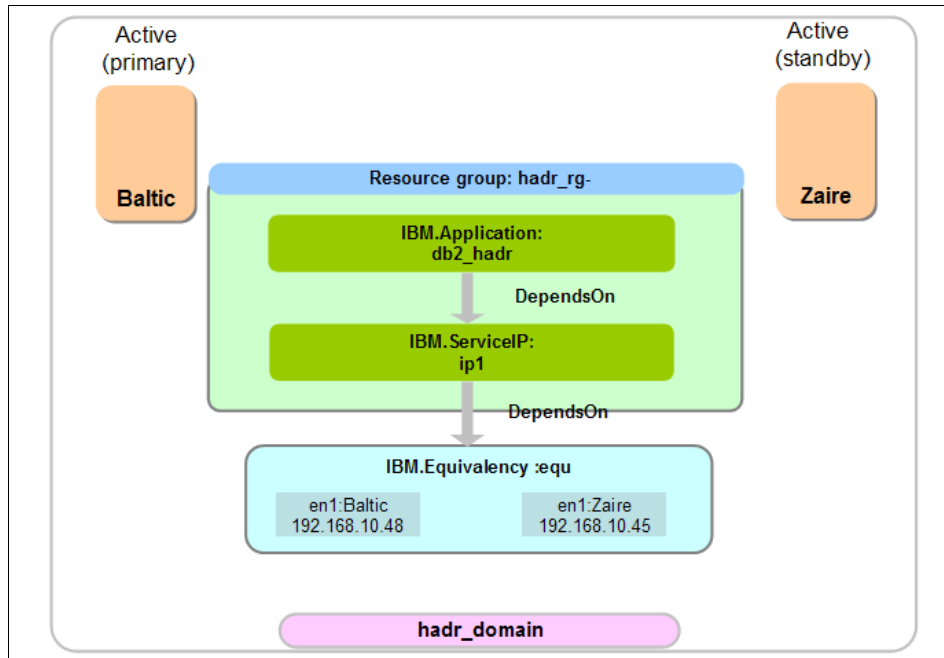


Figure 11-21 Resource groups topology of HADR cluster

## Delete the cluster domain

If you have to delete the cluster domain for any reason, you can execute the following procedures:

1. Execute the **chrg** command on one of the nodes to stop the resource group as follows.
 

```
(P) #chrg -o offline hadr_rg
```
2. Execute the **rmrpdomain** command on one of the nodes to delete the cluster domain.

The **rmrpdomain** command removes the entire TSA configuration in a system and deletes all resources in the cluster. If the other instances are using the domain at the time, the domain is deleted as well. The command helps remove all the residues left from the previous build and gives a clean start.

- ```
(P) #rmrpdomain
(P) #lsrpdomain
```

**Note:** The `rmrpdomain` command leaves the DB2 instances and the HADR replication unaffected. That is, it does not stop the DB2 instances of HADR replications. However, any IP addresses that were made highly available are removed and no longer presented after the `rmrpdomain` command completes.

## Preparing scripts for TSA

To automate HADR takeover with TSA, we utilize the TSA start, monitor, and stop scripts. By including the HADR commands in these scripts, TSA can handle HADR operations in connection with the resource group. You can download the scripts from the following Web site:

<ftp://ftp.software.ibm.com/software/data/db2/express/linuxvalidate/db2salinux.tar.gz>

The scripts for HADR can be placed in the directory of your choice. In this section, we use the customized start, monitor, and stop scripts. We place our scripts in the `/usr/sbin/rsct/sapolicies/hadr/` directory. Appendix C, “TSA scripts” on page 825 lists the customized script samples.

Next we describe some of these TSA controlled scripts and explain how they work.

### ***Start script - `hadr_start.ksh`***

We define a shell script `hadr_start.ksh` as the start script for TSA. This script is executed on the primary node in the following situations when the primary node acquires the resource group:

- ▶ Normal start-up of TSA on the primary node in the cluster
- ▶ Unplanned failover that is triggered by TSA failure detection
- ▶ Planned takeover by moving the resource group with a TSA administrative command

Figure 11-22 shows the flow chart of the script, `hadr_start.ksh`. The script begins with checking whether the HADR role is primary or standby on the node that is executing the script.

If the node is an HADR standby, then the script checks whether the script starting trigger is an unplanned failover or a planned takeover. If the starting trigger is an unplanned failover, which means that it is triggered by failure detection of TSA, then this script issues the `takeover hadr` command with `by force` option.

**Note:** Forcing a takeover can lead to data loss when the HADR state is *NOT* Peer. To prevent data loss, use the *peer\_window* database configuration parameter. Or you can check if the HADR state is *Peer* using the **db2pd** command or checking the message or db2diag.log before forcing a takeover.

Refer to 6.1.1, “Basic configuration parameters” on page 144 for details about *peer\_window*.

On the other hand, if the starting trigger is a planned takeover, which means that it is triggered by moving the resource group with an administrative command of TSA, then this script issues the **takeover hadr** command without the **force** option.

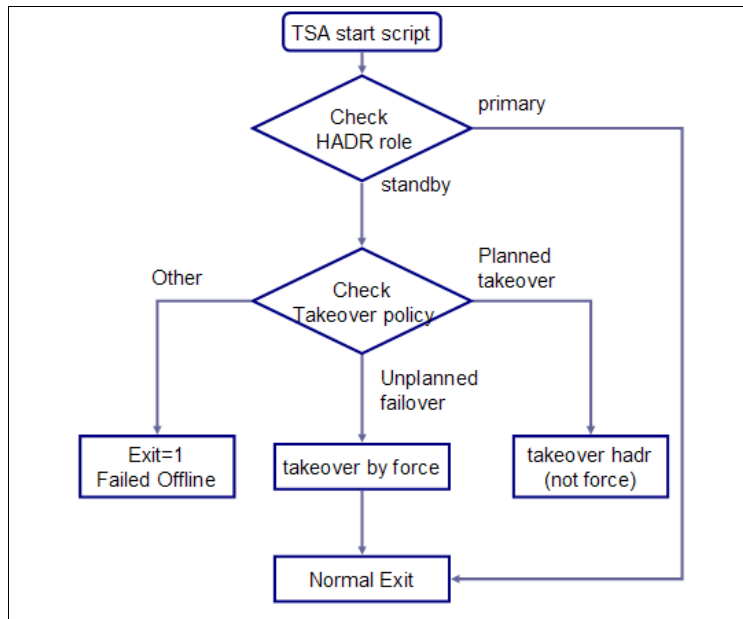


Figure 11-22 *had\_r\_start.ksh* script flow

**Note:** In this case, we use the two flag files to judge the normal start, unplanned failover, or planned takeover. One of the flag files is the `/tmp/hadr_rg_start.flag`, which is created in `had_r_start.ksh` after starting the resource group normally. The other flag file is the `/tmp/planned_tko_flag`, which is created in `planned_takeover.ksh` for planned takeover.

### Monitor script - *hadr\_monitor.ksh*

We define *hadr\_monitor.ksh* as the monitor script for TSA. Figure 11-23 shows the script logic flow. This script is executed on both primary and standby nodes when the cluster domain of TSA starts — that is, when you start the operating system.

This script checks for the existence of the DB2 instance and HADR role and reports the status of the resources back to TSA. If the primary node is on FailedOffline status and the standby node is on Offline status, which means that the primary instance is not alive and the standby instance is normal, TSA executes unplanned failover.

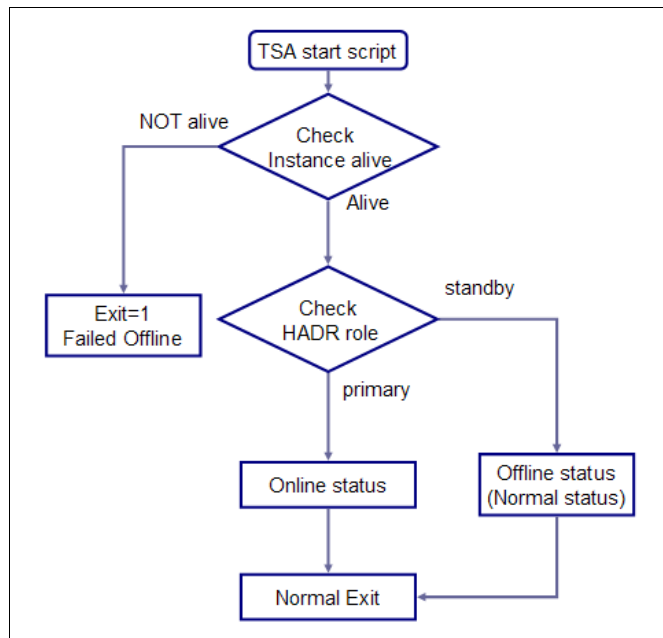


Figure 11-23 *hadr\_monitor.ksh* script flow

### Stop script: *hadr\_stop.ksh*

We define a shell script *hadr\_stop.ksh* as the stop script for TSA. This script is executed when the resource group becomes the offline state on the primary node in the following situations:

- ▶ Normal stop TSA on the primary node in the cluster.
- ▶ Unplanned failover that is triggered by TSA failure detection.
- ▶ Planned takeover by moving the resource group with a TSA administrative command.

This script just removes the flag file /tmp/hadr\_rg\_start.flag, which is created in `hadr_start.ksh` after starting the resource group normally.

### 11.3.3 Administration of HADR with TSA

From time to time, you have to plan a system outage for some maintenance activities such as a software upgrade or DB2 instance recycle for changing non-dynamic database manager parameters. In this section, we demonstrate how to perform some DB2, OS, and TSA operations manually for the planned outage or maintenance.

When you complete the DB2 HADR with TSA setup, you can perform these operations as some simple tests to verify your configuration.

#### Manual stop and start operations

We discuss DB2, OS, and TSA operations in this section for the planned outage or maintenance.

#### ***Stop the standby node and reintegrate it***

If you need to stop TSA on the standby node for maintenance, you can execute the following procedure:

1. Exclude the standby node (Zaire) from the cluster domain using the `samctrl` command as follows on the primary node:

```
samctrl -u a nodename  
(P) #samctrl -u a Zaire
```

Issue the `lssam` command to observe the state of the resources on the primary node. You can see that Zaire is an excluded node, as shown in Example 11-26.

*Example 11-26 lssam output after lssamctrl*

---

```
(P) #lssam  
Online IBM.ResourceGroup:hadr_rg Nominal=Online  
  |- Online IBM.Application:db2_hadr  
    |- Online IBM.Application:db2_hadr:Baltic  
    '- Offline IBM.Application:db2_hadr:Zaire Node=Excluded  
  '- Online IBM.ServiceIP:ip1  
    |- Online IBM.ServiceIP:ip1:Baltic  
    '- Offline IBM.ServiceIP:ip1:Zaire Node=Excluded
```

---

2. Issue the **stoprnode** command as root user from the primary node to stop TSA on the standby. If you execute the **stoprnode** command to stop TSA, you cannot issue any TSA or RSCT operations on the standby node after that. So you must issue this command on the primary node:

```
stoprnode hostname
(P) #stoprnode Zaire
(P) #lsrnode
Name OpState RSCTVersion
Zaire Offline 2.4.8.3
Baltic Online 2.4.8.3
```

3. Issue the **db2stop force** command on the standby instance to stop DB2 instance. You can issue **db2stop** and **db2start** command on the standby node without impacting the activities taking place at the primary database.

```
(S) $db2stop force
```

4. At this point, you can perform the maintenance operations on the standby node, such as reboot OS.
5. To resume the course, issue the **db2start** command on the standby instance.

```
(S) $db2start;db2 activate db sample
```

6. Check the status of the HADR database on both nodes. The primary database should be running on Baltic and the standby database be running on Zaire:

```
(P)(S) $db2pd -hadr -db sample
```

7. Issue the **startprnode** command from the primary node to start TSA on the standby node.

```
startprnode hostname
(P) #startprnode Zaire
(P) #lsrnode
Name OpState RSCTVersion
Zaire Online 2.4.8.3
Baltic Online 2.4.8.3
```

8. To add the standby node back to the cluster domain, issue the **samctrl** command as follows. The output of **lssamctrl** in Example 11-27 shows that the **ExcludedNodes** is **NULL**; nothing is excluded.

```
(P) #samctrl -u d Zaire
```

*Example 11-27 Zaire is no long excluded.*

---

```
(P) #lssamctrl
Displaying SAM Control information:
SAMControl:
```



```

        TimeOut           = 60
        RetryCount        = 3
        Automation        = Auto
        ExcludedNodes     = {}
        ResourceRestartTimeOut = 5
        ActiveVersion     = [2.2.0.5,Thu Apr 24 09:27:52 PDT
2008]
        EnablePublisher   = Disabled
        TraceLevel        = 31

```

---

The output of `lssam` should confirm that confirm the DB2 instance resource on Zaire is online.

## Controlled takeover

The controlled takeover means to manually move a resource group from the primary node to the standby node. The steps are as follows:

1. To move the `hadr_rg` resource group from the primary node (Baltic) to the standby node (Zaire), as a root, run the script `/usr/sbin/rsct/sapolicies/hadr/planned_takeover.sh` from the standby node (Zaire). The script is provided in Appendix C, “TSA scripts” on page 825. This script issues the `rgreq` command.

```

rgreq -o move hadr_rg
(S) #/usr/sbin/rsct/sapolicies/hadr/planned_takeover.sh

```

In this case, we add the logic for both planned and unplanned takeover in the scripts, so we use the scripts to move resource group.

2. As root, check if the standby node (Zaire) has successfully taken over the database (SAMPLE). Example 11-28 shows the `lssam` output indicating that Zaire now is online owning the resources.

*Example 11-28 lssam output after moving the resource group*

---

```

(P) (S) #lssam
Online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Online IBM.Application:db2_hadr
    |- Offline IBM.Application:db2_hadr:Baltic
    '- Online IBM.Application:db2_hadr:Zaire
  '- Online IBM.ServiceIP:ip1
    |- Offline IBM.ServiceIP:ip1:Baltic
    '- Online IBM.ServiceIP:ip1:Zaire

```

---

3. To return the resource group back to the new standby node (Baltic), run the `planned_takeover.sh` script on the new standby:

```

(S) #/usr/sbin/rsct/sapolicies/hadr/planned_takeover.sh

```

### **Stop the both standby and primary node and reintegrate them**

If you need to stop TSA on all nodes for maintenance, you can execute the following procedures:

1. To stop the resource group (*hadr\_rg*), issue the **chrg** command on one of the nodes.

```
chrg -o offline resourcegroupname
#chrg -o offline hadr_rg
```

2. As the root user, use the **stoprpdomain** command to stop the cluster domain (*hadr\_domain*):

```
stoprpdomain domainname
(P) #stoprpdomain hadr_domain
(P) #lsrpdomain
Name          OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
hadr_domain  Offline  2.4.8.3             No              12347   12348
```

3. Stop the DB2 instance on both the primary and the standby nodes:

```
(P)(S) $db2stop force
```

4. Now you can perform the maintenance operations on both primary and standby nodes.

5. To resume to the course, issue the **db2start** command on the standby and primary instance.

```
(P)(S) $db2start;db2 activate db sample
```

6. You check the status of the HADR database on both the nodes. Now the primary database is running on Baltic and the standby database is running on Zaire:

```
(P)(S) $db2pd -hadr -db sample
```

7. Issue the **starttrpdomain** command to start the cluster domain again.

```
starttrpdomain domainname
(P) #starttrpdomain hadr_domain
(P) #lsrpdomain
Name          OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
hadr_domain  Online   2.4.8.3             No              12347   12348
```

8. To start the resource group on the cluster domain, use the **chrg** command:

```
(P) #chrg -o online hadr_rg
```

9. Use the **Issam** command to observe the state of the resources and confirm that the resource of instance on Zaire is online.

### 11.3.4 Failure test of HADR with TSA

In this section we simulate some system failures to verify that the standby node will take over the resources and continue provide service in the event of a system failure. For all the following test cases, we assume that is the primary node is Baltic running the SAMPLE database, and all of the instance resource groups are online.

#### Primary node crash

We discuss node failure by simulating the node crash on the primary node. The node crash simulation can be down with power off or shut down the system.

Figure 11-24 shows the cluster behavior in the event of a primary (service) node crash.

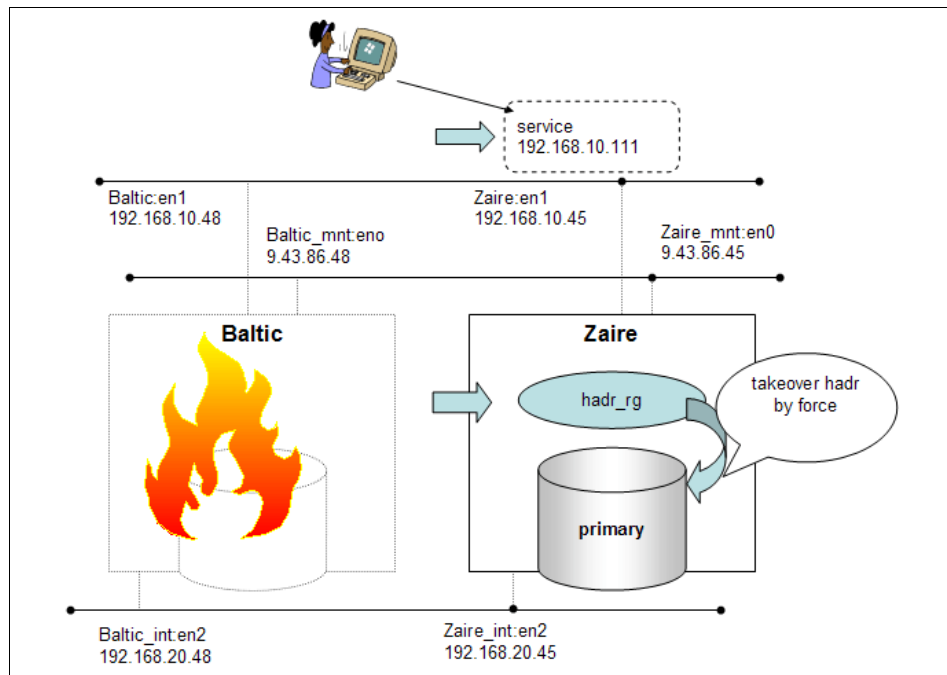


Figure 11-24 Automatic failover to standby node

We power off the primary node and issue the `lssam` command to examine the state of the resources. The resources states are shown in Example 11-29. All resources on the old primary node (Baltic) assume the *Failed Offline* state.

*Example 11-29 Issam output after primary node crash*

---

```
(S) #1ssam
Offline IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Offline IBM.Application:db2_hadr
    |- Failed offline IBM.Application:db2_hadr:Baltic
    '- Offline IBM.Application:db2_hadr:Zaire
  '- Offline IBM.ServiceIP:ip1
    |- Failed offline IBM.ServiceIP:ip1:Baltic
    '- Offline IBM.ServiceIP:ip1:Zaire
```

---

The clients cannot connect to the database. TSA detects the primary nodes outage and starts the failover process. The resource group `hadr_rg` is acquired by the standby node (Zaire):

- ▶ The standby node pings and acquires quorum.
- ▶ The virtual IP address (192.168.10.111) is assigned to the `en1` NIC on the standby node.
- ▶ The TSA Start script, which is related to the resource group, is issued on the standby node. The script includes the **takeover hadr by force** command to change the role of the standby database to the primary.

Example 11-30 shows the resources state after the failover. The resource group `hadr_rg` is online on the new primary (Zaire). The resources on the old primary node (Baltic) assume the Failed Offline state.

*Example 11-30 Issam output after primary node crash*

---

```
(S) #1ssam
Online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Online IBM.Application:db2_hadr
    |- Failed offline IBM.Application:db2_hadr:Baltic
Node=Offline
  '- Online IBM.Application:db2_hadr:Zaire
  '- Online IBM.ServiceIP:ip1
    |- Failed offline IBM.ServiceIP:ip1:Baltic Node=Offline
    '- Online IBM.ServiceIP:ip1:Zaire
```

---

Clients who address the service address succeed in connecting to the new primary database on the surviving node (Zaire), which has the TSA resource group now.

### ***Reintegrating the node***

Figure 11-25 shows the process required to reintegrate the old primary node into clusters.

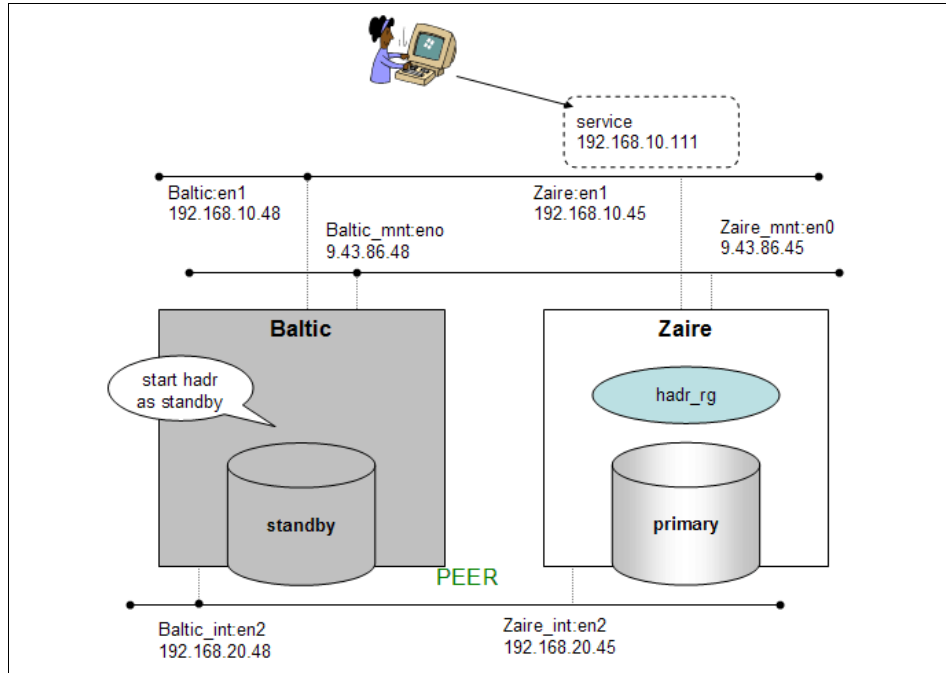


Figure 11-25 Reintegration of old primary database

After the old primary (Baltic) is recovered from the crash, you can start the DB2 instance up and start the database as the HADR standby database on this node by manually executing the **start hadr** command with the **as standby** option.

```
(Baltic) $db2start
(Baltic) $db2 start hadr on db sample as standby
```

The standby database automatically catches up the log records that are processed only on the new primary database during the time that the old primary node is out of order. After the standby database catches up all the log gaps, the HADR primary and standby again return to the Peer state. Reintegration of the old primary database is hereby complete.

Example 11-31 shows the state after reintegration.

Example 11-31 *Issam output after catching up the standby*

```
(P) (S) #lssam
Online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Online IBM.Application:db2_hadr
    |- Offline IBM.Application:db2_hadr:Baltic
    '- Online IBM.Application:db2_hadr:Zaire
```

```
'- Online IBM.ServiceIP:ip1
  |- Offline IBM.ServiceIP:ip1:Baltic
  '- Online IBM.ServiceIP:ip1:Zaire
```

---

## DB2 instance failure

One way to simulate the DB2 instance failure is by terminating the DB2 instance process. We issue the **db2\_ki11** command on the primary instance. All DB2 clients cannot connect the database.

Example 11-32 shows that the HADR resource is in the Pending Online state initially.

*Example 11-32 Issam output after terminating DB2*

---

```
(P)(S) #lssam
Pending online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Pending online IBM.Application:db2_hadr
    |- Failed offline IBM.Application:db2_hadr:Baltic
    '- Pending online IBM.Application:db2_hadr:Zaire
  '- Online IBM.ServiceIP:ip1
    |- Offline IBM.ServiceIP:ip1:Baltic
    '- Online IBM.ServiceIP:ip1:Zaire
```

---

TSA executes a failover and moved the virtual IP address to the standby node. The takeover operation causes the standby database to assume the primary role.

After the failover, the HADR resource group (hadr\_rg) comes back in Online state. But the HADR resource on the old primary node has the Failed Offline state. See Example 11-33.

*Example 11-33 Issam output after the takeover*

---

```
(P)(S) #lssam
Online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Online IBM.Application:db2_hadr
    |- Failed offline IBM.Application:db2_hadr:Baltic
    '- Online IBM.Application:db2_hadr:Zaire
  '- Online IBM.ServiceIP:ip1
    |- Offline IBM.ServiceIP:ip1:Baltic
    '- Online IBM.ServiceIP:ip1:Zaire
```

---

To recover the old primary (Baltic), start the instance in first using **db2start**. Then execute the **start hadr** command with **as standby** on the old primary:

```
db2 start hadr on db dbname as standby
(Baltic) $db2start
(Baltic) $db2 start hadr on db sample as standby
```

Catching-up occurs automatically and the old primary database eventually assumes the new standby role. After that, the system has reached the Peer state. See Example 11-34.

*Example 11-34 Issam output after catching up the old primary*

---

```
(P)(S) #lssam
Online IBM.ResourceGroup:hadr_rg Nominal=Online
  |- Online IBM.Application:db2_hadr
    |- Offline IBM.Application:db2_hadr:Baltic
    '- Online IBM.Application:db2_hadr:Zaire
  '- Online IBM.ServiceIP:ip1
    |- Offline IBM.ServiceIP:ip1:Baltic
    '- Online IBM.ServiceIP:ip1:Zaire
```

---

## 11.4 Automating HADR takeover with TSA on Linux

In DB2 V8.2 and V9.1, clustering software is not automatically integrated with the DB2 product. Instead, users must manually combine the two with appropriate scripts for the clustering software to correctly monitor and manage DB2 HADR resources.

This section illustrates scenarios on the Linux platform. We demonstrate the setup and testing of TSA clustered HADR with DB2 Version 9.1, where TSA is not supplied with the DB2 package, and does not provide the integrated High Availability Feature released in DB2 V9.5. The configuration of DB2 9.5 HADR with TSA is described in Chapter 12, “Configuring clusters using the DB2 9.5 High Availability Feature” on page 477.

### 11.4.1 Setting up HADR with TSA

Here we closely examine the step-by-step creation of a Tivoli System Automation (TSA) cluster managed pair of DB2 HADR nodes. The two major components of this are the TSA Base Component, and DB2.

Prerequisites for the TSA Base Component, and steps describing its installation can be found in Chapter 1 of the *System Automation for Multiplatforms Version 2.2 Installation and Configuration Guide*, SC33-8273.

System requirements for all available versions of DB2 can be found here:

<http://www.ibm.com/software/data/db2/9/sysreqs2.html>

Setting up DB2 HADR in a TSA managed cluster can be done with DB2 Enterprise Server Edition V8.2 or later — we are using version 9.1 in our test environment.

We are assuming an architecture of two servers with SUSE® Linux Enterprise Server x64 10 SP1, one or two physical network interfaces per server with open network connectivity between the two servers, and a separate IP address on a gateway device that is commonly accessible from both servers, to be used as a tiebreaker.

We concentrate on the basic functionality of a simple cluster managing the DB2 instance and the role of each HADR database in the pair. While it is possible to configure a much more complex cluster with more resources and dependencies, we have not included any more than the minimum required resources for the cluster to be functional. Information on adding further resources can be found in the *Tivoli System Automation for Multiplatforms Base Component Administrator's and User's Guide*, SC33-8272. Other cluster resources might include shared disk file systems where key DB2 resources are located, but because we are describing DB2 HADR, where there is no shared disk, this is not a necessary consideration for our cluster definition.

As a general guide, planning for a cluster involves understanding what resources your cluster needs to know about, and how the “Online” or “Offline” state of those resources might affect how the cluster manager behaves. For DB2 HADR, which exists on a two-node cluster in what is termed a “peer domain”, the cluster manager primarily needs to know about the state of the following items:

- ▶ **DB2 instances:** One resource is defined for each instance on each node, and these are added to a resource group—one for each node.
- ▶ **DB2 network communications:** With HADR, DB2 needs to communicate between the two instances as well as with remote client requests. Depending on the number of physical network adapters which DB2 is using to achieve these communications, cluster resources would be defined for each physical network adapter—with equivalency resources defined to enable high availability of multiple network adapters where appropriate. A service IP address (Virtual IP address definition) is often defined in a cluster to act as a single point of external communication for remote clients to communicate with the currently active DB2 database, that is, the one with the HADR primary role in this context.



- ▶ **DB2 HADR database pair:** One resource is defined for each database, and these are combined into a resource group which the cluster manager uses to ensure that only one of these databases can be online, in an HADR primary role.
- ▶ **Tiebreaker resource:** To achieve what is described as a state of “Quorum”, the cluster manager must be satisfied that a majority of nodes have resource dependencies being met, in order to allow availability of the DB2 HADR primary database for remote connection requests. As a rule, the quorum node(s) will have an Online DB2 database, and the other node(s) will set that database state as Offline, to avoid a state of resource contention and data corruption. In a cluster with an even number of nodes, a separate resource must be defined as a proxy, or tiebreaker.

This means that as well as monitoring local resources, and communicating with the other node, the cluster manager running on both sides of the cluster pair issues a heartbeat to the tiebreaker resource, to affirm a means of achieving quorum in the event of resource failures. In some platforms, TSA uses a shared disk resource, which both cluster nodes are usually guaranteed to be able to reach; in others, (such as the Linux platform we are using for our test environment), a network resource is defined. This is literally just an IP address that both cluster nodes are usually able to ping using Internet Control Message Protocol (ICMP).

Detailed information on planning a customized cluster architecture with considerations to meet your own site’s outage risk minimization requirements can be found in the *Tivoli System Automation for Multiplatforms Base Component Administrator’s and User’s Guide*, SC33-8272.

When the definitions are complete, our peer-domain cluster configuration consists of the following definitions:

- ▶ A domain named hadr91.
- ▶ Two domain nodes: These represent our servers: lead and lochnese.
- ▶ A tiebreaker: This separate resource is defined in the form of an IP address, to which both cluster nodes in the peer-domain can connect.
- ▶ Two network IP equivalencies: Each equivalency represents the IP address(es) usable by DB2 on that node for communications. The equivalency is mapped to a virtual IP address used for external communications to DB2.
- ▶ Three resource groups, each with two resources:
  - The first resource group is for one node, initially assigned the primary role. This resource group contains one resource to control the DB2 instance, and one resource to control the IP address equivalency for that node.

- The second resource group is for the second node, initially assigned the standby role, also containing one resource for the DB2 instance, and one for the IP equivalency.
- Both of the first two resource groups should nominally remain in an online operational state.
- The third resource group is critical to our HADR configuration, because it manages which database should have the primary role, and which should have the standby. This is the group used to monitor the current operational HADR state of those databases to make decisions affecting the nominal online or offline operational state of the other two resource groups.

Figure 11-26 shows the relationship between these cluster domain objects in a graphic form.

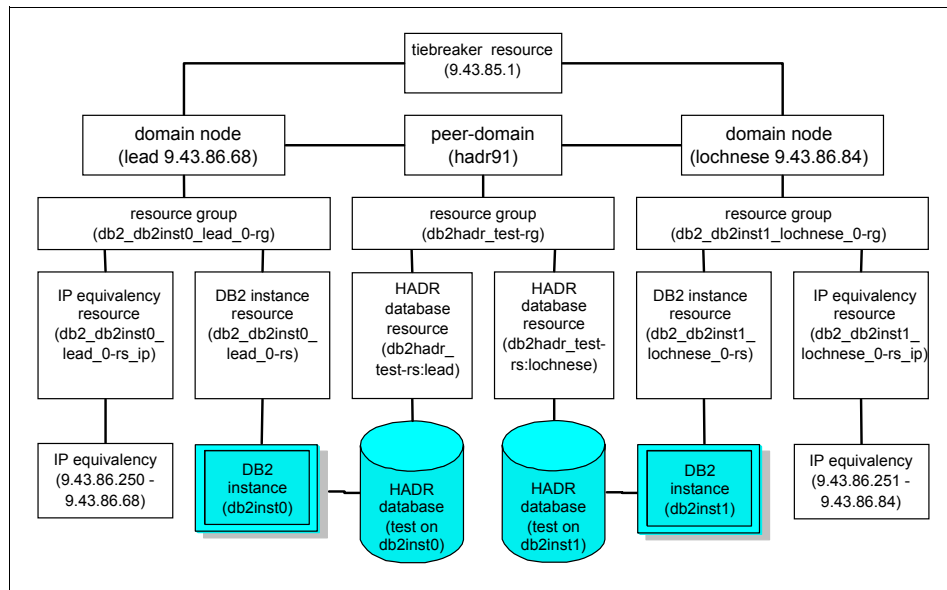


Figure 11-26 Graphic representation of the objects in our example cluster domain

HADR requires a special form of cluster management logic, in that it requires a shared-nothing approach. There is no shared storage, so there are no cluster resources or resource groups defined to control any mounted file systems. This also results in a special form of online-online resource group operational state, which for clusters is otherwise only seen in mutual takeover scenarios. While the nominal operational state of the HADR database pair is reported by cluster manager commands as online-offline, it can be seen with DB2 interrogative commands (such as db2pd) that this is only an abstract cluster representation of the healthy DB2 HADR primary-standby roles.

Planned architecture and naming conventions are laid out in Table 11-1.

*Table 11-1 HADR with TSA using DB2 9.1 on Linux*

| <b>Server</b>                         | <b>lead</b>   | <b>lochnese</b> |
|---------------------------------------|---------------|-----------------|
| IP bond0 (eth0 and eth1 bonded)       | 9.43.86.68    | 9.43.86.84      |
| Virtual IP for equivalency definition | 9.43.86.250   | 9.43.86.251     |
| Virtual IP for service address        | 9.43.86.252   | 9.43.86.252     |
| Gateway tiebreaker                    | 9.43.85.1     | 9.43.85.1       |
| Subnet mask                           | 255.255.252.0 | 255.255.252.0   |
| SVCENAME                              | 55000         | 55000           |
| Local instance                        | db2inst0      | db2inst1        |
| SYSADM_GROUP                          | db2iadm1      | db2iadm1        |
| Local DB name                         | TEST          | TEST            |
| Remote node                           | lochnese      | lead            |
| Remote DB alias                       | TESTLOCH      | TESTLEAD        |
| HADR database role                    | PRIMARY       | STANDBY         |
| HADR_LOCAL_HOST                       | lead          | lochnese        |
| HADR_LOCAL_SVC                        | 55101         | 55102           |
| HADR_REMOTE_HOST                      | lochnese      | lead            |
| HADR_REMOTE_SVC                       | 55102         | 55101           |
| HADR_REMOTE_INST                      | db2inst1      | db2inst0        |
| HADR_TIMEOUT                          | 120           | 120             |
| HADR_SYNCMODE                         | SYNC          | SYNC            |
| DFTDBPATH                             | /db2/data     | /db2/data       |
| LOGARCHMETH1                          | /db2/archlog  | /db2/archlog    |
| Backup path                           | /db2/backup   | /db2/backup     |

Next, we list our assumptions of the state of our test environment prior to configuration of HADR with TSA:

- ▶ DB2 product code has been installed — DB2 9.1 Fix Pack 3 in /opt/ibm/db2/V9.1 for our example.

- ▶ Tivoli System Automation for Multi-Platforms (TSAMP) Base Component product code has been installed — V2.2 Fix Pack 3 in /opt/IBM/tsamp/sam for our example. Information on how to install the TSA Base Component can be found in the TSAMP Installation and Configuration Guide:

[http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8273-01/en\\_US/PDF/HALICG01.pdf](http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8273-01/en_US/PDF/HALICG01.pdf)

- ▶ Both DB2 instance IDs have been created on both nodes and had passwords assigned. We use two different DB2 instance IDs in our test to enable use of the TSA-HADR scripts provided in DB2 V8.2, just in case you are unable to try TSA with DB2 V9.1. There is a restriction in the DB2 V8.2 TSA-HADR scripts that requires the DB2 instance users on each node to be different, but this restriction has been removed in the DB2 V9.1 scripts. Background on this restriction is specially noted in “DB2 HADR configuration” on page 411.

We use the default provided TSA HADR sample scripts from the /opt/ibm/db2/V9.1/ha/tsa/ subdirectory for our test environment.

TSA cluster commands require that both DB2 instance IDs are defined on both cluster nodes. If you are using the DB2 V9.1 scripts, then only the single same DB2 instance ID needs to be defined on each cluster node.

- ▶ DB2 file systems have been created with full permissions granted to the DB2 instance owner and group.
- ▶ DB2 instances have not been created yet.
- ▶ Secure Shell (SSH) daemon **sshd** is running on both servers. Our test environment uses **OpenSSH** version 4.2, as supplied with SuSE Linux Enterprise Server x64 10 SP1.
- ▶ All relevant ports are open on firewall, or no firewall between proposed cluster nodes, and there is open communication on DB2 ports to remote clients.
- ▶ /etc/services file has been populated on each server with the reserved port numbers shown in Table 11-1 on page 409 for the SVCENAME and HADR\_LOCAL\_SVC database manager and database configuration parameters.

We have split the configuration of our clustered environment up into discrete sections:

- ▶ **DB2 HADR configuration:** Optional if you have a suitable existing HADR database pair.
- ▶ **Network configuration:** Recommended if you have more than one physical network adapter on each server.
- ▶ **Secure Shell or Remote Shell configuration:** Recommended — if you do not care about Secure Shell (SSH) communications between the two servers, Remote Shell (rsh) is also covered.

- ▶ **TSA cluster configuration:** Essential steps to set up a peer domain cluster for DB2 HADR.

Given the forgoing list of assumptions, we can now move on to the first set of configuration steps.

## **DB2 HADR configuration**

The TSA Base Component can be installed, and a cluster defined, on top of an existing pair of DB2 HADR databases. You can skip over this configuration step if you already have a suitable HADR database pair ready.

In this step we show you the commands and output from the creation of our DB2 instances, right up until having an active HADR database pair. We do this as briefly as possible to illustrate that configuration of an HADR database pair is a straightforward task, even on the command line, rather than through the DB2 Control Center HADR Configuration wizard. It is also a useful reference point to show where the values in Table 11-1 on page 409 are actually being used in our configuration steps.

**Note:** Perhaps you noticed that our naming for DB2 instances in Table 11-1 on page 409 has different user IDs on each server. This is primarily due to a restriction in the versions of TSA scripts as supplied with DB2 V8.2, which cannot cope with both DB2 instances in an HADR relationship having the same user ID.

A roll-on effect of that restriction is that any table space containers with explicit path names in their definition have to match on primary and standby nodes, and thus cannot have the DB2 instance or other unique identifiers as part of the explicit path name. Table space containers using relative path names (the default method) do not have this restriction.

We are taking these legacy requirements into account for our example only to satisfy the possibility of users relying on the scripts supplied with DB2 V8.2. These restrictions have been removed with the scripts supplied with DB2 V9.1, however, those scripts might not work seamlessly with DB2 V8.2 TSA HADR clusters without user customization.

Our assumptions included having created the DB2 instance user IDs, but not the instances themselves. The following step carries out that work.

### ***Create a DB2 instance***

Create a DB2 instance (and optionally a DB2 Admin Server) for each cluster node. Example 11-35 shows the command we use to create instances in both nodes.

*Example 11-35 Creation of DB2 instances for each user*

---

```
lead:~ #/opt/ibm/db2/V9.1/instance/db2icrt -p 55000 -u db2fenc1
db2inst0
DBI1070I Program db2icrt completed successfully.
lead:~ # /opt/ibm/db2/V9.1/instance/dascrt -u db2as
SQL4406W The DB2 Administration Server was started successfully.
DBI1070I Program dascrt completed successfully.

lochnese:~ #/opt/ibm/db2/V9.1/instance/db2icrt -p 55000 -u db2fenc1
db2inst1
DBI1070I Program db2icrt completed successfully.
lochnese:~ # /opt/ibm/db2/V9.1/instance/dascrt -u db2as
SQL4406W The DB2 Administration Server was started successfully.
DBI1070I Program dascrt completed successfully.
```

---

As with any standard creation of an HADR database, it is created first on the primary server, configured with archive logging, and HADR db cfg parameters are set. We use an empty basic database for our environment, called TEST.

The database on this primary server is backed up (online backup to allow a database restore with rollforward pending mode) and taken across to the standby server through one of a variety of methods. For our example, we simply **scp** the backup file across to a subdirectory on the standby server.

On the standby server, the database is restored without rolling forward, and the reciprocal db cfg parameters for the standby are configured to correctly refer back to the primary HADR database and instance.

The HADR database can then be started as standby on the standby server, and then started as primary on the primary server, followed by a **db2pd** command to verify the status of the HADR pair as Connected and in Peer state.

All these steps are shown in Example 11-36, beginning with registry variable settings and update dbm cfg steps executed just after the DB2 instances were created on each node.

*Example 11-36 Setting up a generic HADR database pair for our test environment*

---

```
###On Primary server:
db2inst0@lead:~> db2set DB2COMM=tcPIP
db2inst0@lead:~> db2 update dbm cfg using DFTDBPATH /db2/data
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst0@lead:~> db2 update dbm cfg using SVCENAME 55000
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
```

```

db2inst0@lead:~> db2gcf -d -i db2inst0

Instance : db2inst0
DB2 Stop : Success
db2inst0@lead:~> db2gcf -u -i db2inst0

Instance : db2inst0
DB2 Start : Success
db2inst0@lead:~> db2 create db test
DB20000I The CREATE DATABASE command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_LOCAL_HOST lead
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_LOCAL_SVC 55101
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_REMOTE_HOST lochinese
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_REMOTE_SVC 55102
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_REMOTE_INST db2inst1
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_TIMEOUT 120
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 update db cfg for test using HADR_SYNCMODE SYNC
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

db2inst0@lead:~> db2 update db cfg for test using LOGARCHMETH1
disk:/db2/archlog
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, all
applications must disconnect from this database before the changes become
effective.
db2inst0@lead:~> db2 update db cfg for test using logindexbuild on
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, all
applications must disconnect from this database before the changes become
effective.

db2inst0@lead:~> db2 activate db test
SQL1490W Activate database is successful, however, the database has already
been activated on one or more nodes.
db2inst0@lead:~> db2 deactivate db test
DB20000I The DEACTIVATE DATABASE command completed successfully.
db2inst0@lead:~> db2 backup db test to /db2/backup compress

Backup successful. The timestamp for this backup image is : 20080416144937

```

```

db2inst0@lead:~> db2 activate db test
DB20000I The ACTIVATE DATABASE command completed successfully.
db2inst0@lead:~> rm /db2/backup*.001
db2inst0@lead:~> db2 backup db test online to /db2/backup compress

Backup successful. The timestamp for this backup image is : 20080416150709

db2inst0@lead:~> scp /db2/backup/* db2inst1@lochnese:/db2/backup
TEST.0.db2inst0.NODE0000.CATN0000.20080416150709.001
100% 22MB 11.1MB/s 00:02

###On Standby server:
db2inst1@lochnese:~> db2set DB2COMM=tcPIP
db2inst1@lochnese:~> db2 update dbm cfg using DFTDBPATH /db2/data
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@lochnese:~> db2 update dbm cfg using SVCENAME 55000
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@lochnese:~> db2gcf -d -i db2inst1

Instance : db2inst1
DB2 Stop : Success
db2inst1@lochnese:~> db2gcf -u -i db2inst1

Instance : db2inst1
DB2 Start : Success
db2inst1@lochnese:~> ls /db2/backup
TEST.0.db2inst0.NODE0000.CATN0000.20080416150709.001
db2inst1@lochnese:~> db2 restore db test from /db2/backup
DB20000I The RESTORE DATABASE command completed successfully.
db2inst1@lochnese:~> db2 update db cfg for test using HADR_LOCAL_HOST lochnese
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lochnese:~> db2 update db cfg for test using HADR_LOCAL_SVC 55102
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lochnese:~> db2 update db cfg for test using HADR_REMOTE_HOST lead
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lochnese:~> db2 update db cfg for test using HADR_REMOTE_SVC 55101
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lochnese:~> db2 update db cfg for test using HADR_REMOTE_INST db2inst0
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lochnese:~> db2 start hadr on db test as standby
DB20000I The START HADR ON DATABASE command completed successfully.

### On Primary server:
db2inst0@lead:~> db2 start hadr on db test as primary
DB20000I The START HADR ON DATABASE command completed successfully.
db2inst0@lead:~> db2pd -d test -hadr

```



Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:00:12

HADR Information:

| Role    | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer  | Sync     | 0                | 0                    |

| ConnectStatus | ConnectTime                           | Timeout |
|---------------|---------------------------------------|---------|
| Connected     | Wed Apr 16 15:32:22 2008 (1208385142) | 15      |

| LocalHost | LocalService |
|-----------|--------------|
| lead      | 55101        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| lochnese   | 55102         | db2inst1       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000002.LOG | 0         | 0x00000000017A0000 |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000002.LOG | 0         | 0x00000000017A0000 |

---

As an additional step, remote nodes, database aliases, and the DB2 Automatic Client Reroute (ACR) facility are defined using commands shown in Example 11-37. This step is optional if using a cluster to manage IP address switching. We use ACR in our test environment as an illustration of how it functions in a clustered environment.

*Example 11-37 Setting up the remote node, db & ACR alternate server definitions*

---

```
db2inst0@lead:~> db2 catalog tcpip node lochnese remote lochnese server 55000
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
db2inst0@lead:~> db2 catalog db testloch at node lochnese
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
db2inst0@lead:~> db2 update alternate server for database test using hostname lochnese
port 55000
DB20000I The UPDATE ALTERNATE SERVER FOR DATABASE command completed
successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
```

```
db2inst1@lochnese:~> db2 catalog tcpip node lead remote lead server 55000
DB20000I The CATALOG TCPIP NODE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
db2inst1@lochnese:~> db2 catalog db testlead at node lead
DB20000I The CATALOG DATABASE command completed successfully.
DB21056W Directory changes may not be effective until the directory cache is refreshed.
```

```
db2inst1@lochnese:~/script> db2 update alternate server for database test using hostname  
lead port 55000  
DB20000I The UPDATE ALTERNATE SERVER FOR DATABASE command completed  
successfully.  
DB21056W Directory changes may not be effective until the directory cache is refreshed.
```

---

## Network configuration

We have attempted to make the network configuration of our Linux test environment as compatible as possible for you to base your own environment on. We use fixed IP addresses for each of the two DB2 servers for ease of reference and explanation of which actual network object we are referring to. These can be substituted with dynamic IP addressing if desired.

We use a bonded network definition to support high availability of the network. This makes the TSA cluster definition process the same for environments that only have one physical network adapter/Network Interface Card (NIC) per server. Single NIC environments can skip the bonding process altogether and use their *eth0* adapter definition instead of the *bond0* adapter definition we are using.

### Configuring bonded NIC definition

As a very brief explanation of network bonding, the concept is based on having two or more slave network adapter definitions being controlled by a “bond” type network adapter definition. Various modes and options exist for bonded network definitions. We use the active backup/hot standby mode. This allows for seamless high availability of the slave network adapters, in addition to load sharing functionality. All slave network adapters/NICs appear to the outside world to be a single NIC on a single IP address, which is registered against our host name definition in */etc/hosts*, or in a DNS registry for dynamic IP addressing.

SUSE Linux Enterprise Server x64 10 SP1 supports bonded NIC configuration inside YAST or YAST2. For configuration instructions on other versions of this distribution, the following link contains all required information:

[https://secure-support.novell.com/KanisaPlatform/Publishing/133/3929220\\_f.SAL\\_Public.html](https://secure-support.novell.com/KanisaPlatform/Publishing/133/3929220_f.SAL_Public.html)

For generic information on network bonding with other Linux distributions, this link can be used:

<http://sourceforge.net/projects/bonding/>

If the kernel-source package is installed on your system, the following file will contain the same documentation:

*/usr/src/linux/Documentation/networking/bonding.txt*

## Confirm IP setup

Here we establish that the network adapter definitions for each NIC exist on each server. In our example architecture, there should be an entry for both physical adapters (eth0, and eth1), plus a bonded ethernet adapter definition, and a loopback adapter (lo).

Example 11-38 shows the IP configuration in our Lab on one of our pair of servers. The format is the same for both, but the bond0 adapter inet addr: value is 9.43.86.68 for lead, and 9.43.86.84 for lochnese.

This output should help in comparison with your own systems.

### Example 11-38 IP configuration for one of our servers

---

```
lead:~ # ifconfig
bond0    Link encap:Ethernet  HWaddr 00:02:55:E4:5C:56
         inet addr:9.43.86.68  Bcast:9.43.87.255  Mask:255.255.252.0
         inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
         UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
         RX packets:2773001 errors:0 dropped:0 overruns:0 frame:0
         TX packets:1384408 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:293426353 (279.8 Mb)  TX bytes:523417748 (499.1 Mb)

eth0     Link encap:Ethernet  HWaddr 00:02:55:E4:5C:56
         inet6 addr: fe80::202:55ff:fee4:5c56/64 Scope:Link
         UP BROADCAST RUNNING NOARP SLAVE MULTICAST  MTU:1500  Metric:1
         RX packets:3849 errors:0 dropped:0 overruns:0 frame:0
         TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:346808 (338.6 Kb)  TX bytes:222 (222.0 b)
         Interrupt:169

eth1     Link encap:Ethernet  HWaddr 00:02:55:E4:5C:56
         inet6 addr: fe80::202:55ff:fee4:5c56/64 Scope:Link
         UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
         RX packets:2769152 errors:0 dropped:0 overruns:0 frame:0
         TX packets:1384405 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:293079545 (279.5 Mb)  TX bytes:523417526 (499.1 Mb)

lo       Link encap:Local Loopback
         inet addr:127.0.0.1  Mask:255.0.0.0
         inet6 addr: ::1/128 Scope:Host
         UP LOOPBACK RUNNING  MTU:16436  Metric:1
         RX packets:20876 errors:0 dropped:0 overruns:0 frame:0
         TX packets:20876 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:0
         RX bytes:4109155 (3.9 Mb)  TX bytes:4109155 (3.9 Mb)
```

---

If you have only a single NIC per server, then you should only see an *eth0* and *lo* (loopback) adapter definition as output from each **ifconfig** command.

### **Confirm the /etc/hosts content**

We are using static IP addresses for our test environment, so the */etc/hosts* files on both servers must contain matching host name entries for those two servers, as shown in Example 11-39.

*Example 11-39 /etc/hosts lines in lead and lochnese*

---

```
lead:~ # cat /etc/hosts |grep 9.43
9.43.86.68      lead.itsosj.sanjose.ibm.com lead
9.43.86.84      lochnese.itsosj.sanjose.ibm.com lochnese

lochnese:~ # cat /etc/hosts |grep 9.43
9.43.86.68      lead.itsosj.sanjose.ibm.com lead
9.43.86.84      lochnese.itsosj.sanjose.ibm.com lochnese
```

---

The two nodes should be able to issue **ping** commands to each other, and both should be able to reach the common IP address to be used as a tiebreaker, as confirmed in the steps shown in Example 11-40.

*Example 11-40 Ping to each node from lead and lochnese*

---

```
lead:~ # ping lochnese
PING lochnese.itsosj.sanjose.ibm.com (9.43.86.84) 56(84) bytes of data.
64 bytes from lochnese.itsosj.sanjose.ibm.com (9.43.86.84): icmp_seq=1 ttl=64 time=0.134 ms
64 bytes from lochnese.itsosj.sanjose.ibm.com (9.43.86.84): icmp_seq=2 ttl=64 time=0.118 ms

--- lochnese.itsosj.sanjose.ibm.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.118/0.126/0.134/0.008 ms
lead:~ # ping 9.43.85.1
PING 9.43.85.1 (9.43.85.1) 56(84) bytes of data.
64 bytes from 9.43.85.1: icmp_seq=1 ttl=255 time=0.161 ms
64 bytes from 9.43.85.1: icmp_seq=2 ttl=255 time=0.154 ms

--- 9.43.85.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.154/0.157/0.161/0.013 ms

lochnese:~ # ping lead
PING lead.itsosj.sanjose.ibm.com (9.43.86.68) 56(84) bytes of data.
64 bytes from lead.itsosj.sanjose.ibm.com (9.43.86.68): icmp_seq=1 ttl=64 time=0.126 ms
64 bytes from lead.itsosj.sanjose.ibm.com (9.43.86.68): icmp_seq=2 ttl=64 time=0.114 ms

--- lead.itsosj.sanjose.ibm.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.114/0.120/0.126/0.006 ms
```

```
tochnese:~ # ping 9.43.85.1
PING 9.43.85.1 (9.43.85.1) 56(84) bytes of data.
64 bytes from 9.43.85.1: icmp_seq=1 ttl=255 time=0.164 ms
64 bytes from 9.43.85.1: icmp_seq=2 ttl=255 time=0.144 ms

--- 9.43.85.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.144/0.154/0.164/0.010 ms
```

---

## **Secure Shell or Remote Shell configuration**

We acknowledge that in a tightly secured and firewall isolated network environment, an HADR database pair on the same subnet does not require Secure Shell (SSH) to be used as a cluster remote command mechanism, and that some overhead is involved in the encryption of data sent. In many sites however, Remote Shell (rsh) has been completely disabled for security reasons, so there is no alternative.

In addition to the fact that Remote Shell does not encrypt passwords or data traffic sent across the network, many privilege escalation or attack exploits such as Distributed Denial of Service (DDoS) remain as issues with Remote Shell, usually through some mechanism of buffer overflow.

Unfortunately for secure sites, Remote Shell is still configured as the default remote command mechanism in the TSA HADR scripts, so we have some work to do in order to configure SSH for use by those scripts.

## **Remote Shell**

For those who are allowed to use rsh on their site, the following steps are required to enable its use for the TSA cluster commands:

- ▶ The presence of rsh can be confirmed on any Red Hat Package Manager (rpm)-based Linux distribution by use of the `rpm -qa |grep rsh` command, with results appearing for both rsh and rsh-server packages. (Fedora, SuSE, and Mandriva are all examples of Red Hat Package Manager Linux distributions.)
- ▶ Enable rsh usage with the root user issuing the command: **chkconfig rsh on**
- ▶ On rpm-based Linux distributions, have root add the **rsh** command into the `/etc/securityty` file.
- ▶ Start the rsh service on both DB2 servers by having root restart `inetd` or `xinetd` (depending on your Linux server's network daemon configuration):  

```
/etc/init.d/inetd restart
or
/etc/init.d/xinetd restart
```

- ▶ Confirm the restarted network daemon(s) are running the rsh service with a `chkconfig` command (output should show rsh with a status of “On”):

```
chkconfig --list | grep rsh
```

- ▶ Enable root and DB2 instance users to issue remote commands with an entry for both servers in `/root/.rhosts` on both DB2 servers. For example, our test configuration would require these entries in `/root/.rhosts` on both servers:

```
lead root db2inst0 db2inst1
lochnese root db2inst0 db2inst1
```

- ▶ Verify that rsh remote commands are working by having each user test a command against the remote server. The following example command and output by root indicates a successful test:

```
lead:~ #rsh lochnese hostname
lochnese
```

## Secure Shell

As our preferred solution, we present the following three steps to configure SSH for use in a TSA HADR cluster:

1. Configure password-less SSH
2. Configure DB2 commands to use SSH
3. Configure TSA commands to use SSH

Here are the steps in detail:

1. Configure password-less SSH:

We configure SSH to allow specific users to run remote commands without requiring a password. For our test environment, we run the configuration steps against a total of six users, that is, root, db2inst0, and db2inst1 on both lead and lochnese. While the DB2 instance ID on each node is different, each user ID must also be present on the other server to enable the later use of TSA registration commands.

Certain steps require that the previous steps have been completed for all users, specifically where the unique public key staging files are concatenated back into the `authorized_keys` file for each user. This means that steps a, b, and c could be put in a single script which each user can run. The “Set up inbound `authorized_keys` file:” step should be in a separate script to be run for every user only after the previous steps have been run for every user, and then finally the “Test password-less SSH:” step can be run for every user, also from a script if desired.

The following steps are used to configure password-less SSH:

a. Generate SSH keys:

Generate private and public dsa and rsa keys into the `~/.ssh` directory using the `ssh-keygen` command:

```
/usr/bin/ssh-keygen -f ~/.ssh/id_dsa -q -t dsa -N ""  
/usr/bin/ssh-keygen -f ~/.ssh/id_rsa -q -t rsa -N ""
```

These keys might already exist, and the `ssh-keygen` command will prompt you with a choice to replace the existing key files. In this case, we would prefer not to replace them, because they are most likely already being used for connection to other servers. We would also check at this stage if an `authorized_keys` file exists, and create a backup of it with a `cp` command in case our next set of steps does not result in a clean `authorized_keys` file:

```
cp ~/.ssh/authorized_keys ~/.ssh/authorized_keys_backup
```

Example 11-41 shows the coding for issuing `ssh-keygen` from one of the user IDs to generate an initial `~/.ssh` subdirectory populated with private and public key-pairs. For our test environment, this is performed for `root`, `db2inst0`, and `db2inst1` on both nodes.

*Example 11-41 Generating private and public ssh keys*

---

```
lead:~ # /usr/bin/ssh-keygen -f ~/.ssh/id_dsa -q -t dsa -N ""  
lead:~ # /usr/bin/ssh-keygen -f ~/.ssh/id_rsa -q -t rsa -N ""
```

---

If either of the `dsa` or `rsa` key pairs already exist, a warning message is issued. Our recommendation is to not replace any existing key pairs, especially for the `root` user. Care should also be taken not to destroy any existing `authorized_keys` file in the `root` user `~/.ssh` subdirectory, which is why we recommended creating a backup of that file with a `cp` command at the start of this step.

b. Set up local public key staging file:

Copy the public keys to a staging file in the `~/.ssh` subdirectory. This can be done in a variety of ways. We choose two `cat` commands with redirection; the first `cat` overwrites any previous content and the second concatenates. Example 11-42 shows this being done for one of the `root` users, with a generic command syntax that could be employed for any of the users.

*Example 11-42 Copying to staging file pub\_keys*

---

```
lead:~ # cat .ssh/id_dsa.pub > .ssh/pub_keys  
lead:~ # cat .ssh/id_rsa.pub >> .ssh/pub_keys
```

---

c. Set up outbound public key staging file:

Copy the staging file to a uniquely named staging file for each other user on each server you want the current user to be able to run **ssh** commands against. You most likely do not want to enable password-less remote **ssh** to either of the root users from any of the DB2 instance users. This is the step that requires customization for each individual user if a generic script is not used. A unique staging file must be copied to the remote user's subdirectory so as to avoid overwrites of public key staging files from other users.

If you choose to combine steps a, b, and c into a script, take care that the remote target subdirectory already exists, which is why we are not copying public keys directly to the `~/.ssh` subdirectory of the remote user. Unless a given user has run **ssh-keygen**, or a known hosts file has been generated by running the **ssh** client, the `~/.ssh` subdirectory will not exist yet.

Example 11-43 shows how we use the Secure Copy (**scp**) command in an example script to get the local staging file to the remote locations, and how we rely on environment variables and context testing to avoid giving unwanted accesses to root. This script makes step this step as generically functional as possible for every user, and can be appended to the commands for step a) and b) to make a single script.

---

*Example 11-43 Script file to copy public ssh keys to the other users*

---

```
lh=`hostname`
for u in root db2inst0 db2inst1 ; do
  for h in lead lochinese ; do
    if [ "$USER" != "root" -a "$u" == "root" ] ; then
      echo "skipping nonroot scp to root"
    else
      scp $HOME/.ssh/pub_keys $u@"$h":pub_keys_"$USER_"$lh
    fi
  done
done
```

---

d. Set up inbound `authorized_keys` file:

Concatenate the unique staging files from remote users to the `authorized_keys` file in the local `~/.ssh` directory.

If the script from Example 11-43 is executed by the root user and each DB2 instance user on both nodes, their home directories should now have public key staging files with unique names from the other users, including one from itself. Depending on whether your site allows the DB2 instance users to `su` to the root user without a password, there may be no public key files from the DB2 users in the root user home directory. All we do at



this stage is concatenate each of those public key files into a common local `authorized_keys` file, for each user.

Our script in Example 11-44 does this and removes the obsolete public key staging files. In addition to keeping things tidy, it helps to prevent accidentally running the script more than once and creating redundant entries in the local `~/.ssh/authorized_keys` file. If this does somehow happen, the bloated `authorized_keys` file can be safely deleted or renamed and the contents recreated from the other users' public key files. We recommended in step a on page 421 that an initial backup should have been taken of any pre-existing `authorized_keys` file, especially for the root user.

*Example 11-44 Concatenating staging files into the local `authorized_keys` file*

---

```
for f in $HOME/pub_keys_* ; do
  cat $f >> $HOME/.ssh/authorized_keys
  rm $f
done
```

---

e. Test password-less SSH:

Confirm that the `ssh` command does not require a password to connect to each of the other users. This can be done with a command script run against each user, which shows the expected remote user and hostname as output text. Example 11-45 shows our command script, and the expected output from running it against a DB2 instance user. We expect with our scripts that DB2 instance users will not be able to connect to root users without a password, and we expect a successful test from a loopback `ssh` connection to the same user on the same server without requiring a password.

*Example 11-45 Testing whether `ssh` works without a password*

---

```
for u in root db2inst0 db2inst1 ; do
  for h in lead lochnese ; do
    ssh $u@"$h "whoami;hostname"
    echo " "
  done
done
```

```
-----Expected output from this script when run against any of the
DB2 instance users
Password:
root
lead
```

```
Password:
```

```
root
lochnese
```

```
db2inst0
lead
```

```
db2inst0
lochnese
```

```
db2inst1
lead
```

```
db2inst1
lochnese
```

---

For more details about **ssh** and **ssh-keygen**, we recommend that you refer to the man pages on your Linux distribution.

## 2. Configure DB2 to use SSH:

After having set up SSH password-less remote commands, each DB2 instance must be specifically configured to use SSH rather than RSH for remote commands such as **db2gcf**. This can most easily be accomplished by setting the DB2 registry variable **DB2RSHCMD** to the path where the **ssh** binary is located, and recycling the DB2 instance, as we show in Example 11-46.

*Example 11-46 Setting the DB2RSHCMD registry variable on a DB2 instance*

---

```
db2set -i db2inst1 DB2RSHCMD=/usr/bin/ssh
db2stop
db2start
```

---

After recycling the DB2 instance, this registry variable should take effect for all remote commands issued against that instance.

For background information and your further reference, this URL explains in detail how to set up DB2 to use SSH authentication rather than the old default RSH, as used by commands such as **db2gcf**:

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0506finance/index.html>

The following link in the DB2 V9.1 Information Center sets out the basic requirements for changing DB2 remote commands from using RSH to SSH:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0005828.htm>

### 3. Using TSA with SSH:

When installed into the default location, DB2 V9.1 contains TSA-specific registration and management scripts in:

```
/opt/ibm/db2/V9.1/ha/tsa
```

DB2 V8.2 has these scripts in the default location of:

```
/opt/IBM/db2/V8.2/ha/salinux
```

Because this location is not generally set up in the PATH environment variable, certain scripts such as **regdb2salin** necessitate use of explicit path names.

What we looked for initially in this script subdirectory was any evidence of where **rsh** was being referenced. This investigative work is for your information only, and no action is required at this point. We run a case-insensitive search on “rsh” in the TSA High Availability scripts subdirectory within the DB2 product libraries:

```
/opt/ibm/db2/V9.1/ha/tsa> for a in *;do echo $a;cat $a |grep -i rsh;done
```

Our investigation concludes that only the **regdb2salin** script contains references to **rsh**.

On further inspection, there is a global variables section near the header comments of the **regdb2salin** file, containing the text in Example 11-47 on page 425.

*Example 11-47 Global variable section of regdb2salin*

---

```
# Globals...
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/sbin/rsct/bin
export PATH
VERBOSE=false
SSH_OR_RSH=rsh
SCP_OR_RCP=rcp
UNAME=$(uname)
MYID=root
```

---

This is the stage where you should make changes, either in the source code for **regdb2salin**, or at runtime, as explained next.

By default, the non-secure **rsh** command is set as the remote shell method, and **rcp** as the default remote copy. These can be changed permanently by having the root user ID editing the initial setting in **regdb2salin** on both nodes from:

```
SSH_OR_RSH=rsh
SCP_OR_RCP=rcp
```

to:

```
SSH_OR_RSH=ssh
SCP_OR_RCP=scp
```

The `SSH_OR_RSH` and `SCP_OR_RCP` parameters can also be changed at runtime with use of the `-s` flag to force usage of `ssh` and `scp` commands.

For diagnostic purposes, the `VERBOSE=false` can be changed to `VERBOSE=true`, or just use the `-v` flag as a `regdb2salin` command parameter.

After being registered within a TSA HADR cluster by the `regdb2salin` command, the other scripts are controlled by and executed from the TSA software, so the initial setting to use `ssh` or `rsh` is carried over. For security and integrity purposes, all the DB2 TSA scripts are owned by `root:root`, with permissions of `555 (r-xr-xr-x)`, on the files, and on the parent subdirectory structure, which restricts backing up or editing.

Executing `regdb2salin` in verbose mode sheds light on what commands are being issued if a problem occurs. For example, at the first time an attempt is made to start DB2, this command is executed:

```
ssh lead '/usr/sbin/rsct/sapolicies/db2/db2_start.ksh db2inst0 0'
```

If this step hangs or fails, messages will appear in `db2diag.log`. However, nothing may happen in the `regdb2salin` script.

Inside that `db2_start.ksh` script, this command has been executed:

```
/home/db2inst0/sqllib/bin/db2gcf -t 60 -u -p 0 -i db2inst0
```

Extrapolating those parameters in the command reference section of the DB2 Information Center, we can see that a request is being made to bring partition 0 of the `db2inst0` instance up, with a *timeout* value override of 60 seconds. (Partition numbers are ignored for the `db2gcf` command in a non-partitioned environment).

Due to issues that might otherwise arise later when first executing `regdb2salin`, it is a good idea to have the root user test starting and stopping the DB2 instance with the `db2gcf` command prior to running `regdb2salin`, as shown in Example 11-48.

*Example 11-48 Testing db2gcf command to confirm functionality using ssh*

---

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -s -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 State : Operable
```

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -u -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 Start : Success
```

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -s -i db2inst0
```

```
Instance : db2inst0
DB2 State : Available
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -d -i db2inst0
```

```
Instance : db2inst0
DB2 Stop : Success
```

---

Information on the syntax for the db2gcf command can be found in:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0010986.htm>

One other example of **regdb2salin** failing that we discovered with this scenario of using two different DB2 instance user IDs, was that **regdb2salin** still expected and required both those DB2 instance user IDs to be on both nodes, even if only one of each had an associated DB2 instance created on it. This in turn necessitated sharing all public **ssh** keys in a four-way combination between those two IDs on both nodes. After this was established, **regdb2salin** worked flawlessly, but executing **regdb2salin** in both verbose and non-verbose mode was invaluable in establishing the cause of the error.

A partial or failed execution of **regdb2salin** can be restarted after cleaning up definitions with the **unregdb2salin** command, as shown in Example 11-49 on page 427. Note that this does not remove the network interface equivalencies that were set up in a previous step.

*Example 11-49 Cleaning up a failed regdb2salin execution*

---

```
lead:/opt/ibm/db2/V9.1/ha/tsa # ./unregdb2salin -a db2inst0
Stopping all online resource groups for instance db2inst0...
Removing all relationship's ...
Removing equivalency's ...
Removing all resources ...
Removing all resource groups ...
lead:/opt/ibm/db2/V9.1/ha/tsa # lsequ
Displaying Equivalencies :
virpubnic_lochnese
virpubnic_lead
```

---

This concludes our requirements for setting up SSH to work within DB2 and TSA.

## TSA cluster configuration

Finally, we arrive at the stage of executing TSA cluster configuration commands, as well as RSCT commands to define the underlying peer domain cluster that TSA is automating.

## ***TSA peer domain scope configuration***

To set the scope of the cluster for a peer domain (our two-node cluster is classed as a peer domain), the following coding must be inserted into the root's `/etc/bash.bashrc.local` file if using bash (Bourne Again SHell), or the equivalent for whichever UNIX shell is being used:

```
export CT_MANAGEMENT_SCOPE=2
```

To make this effective, have root log out and back in again, or run the export at the command line before proceeding.

## ***Prepare and start the cluster***

Follow these steps to prepare and start the cluster:

- ▶ Prepare both nodes for the cluster with **preprnode** command. This prepare command authorizes each cluster node for commands to be issued against cluster objects.

```
lochnese:~ # preprnode lead lochnese
lead:~ # preprnode lead lochnese
```

- ▶ Make the cluster domain from one of the nodes with command **mkrpdomain**, specifying the list of servers to be defined as cluster nodes. We call our cluster domain "hadr91", and include the two DB2 HADR servers as cluster nodes.

```
lead:~ # mkrpdomain hadr91 lead lochnese
```

- ▶ Start the cluster domain with command **startdomain**:

```
lead:~ # startdomain hadr91
Starting the domain should bring it into an Online operational state (OpState).
```

- ▶ List the operational state (Online or Offline) of cluster domains with command **lsrpdomain**, to confirm that the domain reaches an OpState of "Online":

```
lead:~ # lsrpdomain
Name      OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
hadr91    Offline  2.5.0.2             No              12347   12348
lead:~ # lsrpdomain
Name      OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
hadr91    Pending online  2.5.0.2             No              12347   12348
lead:~ # lsrpdomain
Name      OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
hadr91    Online   2.5.0.2             No              12347   12348
```

Check that the cluster domain is also shown as "Online" from the other cluster node:

```
lochnese:~ # lsrpdomain
```

```
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
hadr91 Online 2.5.0.2 No 12347 12348
```

- Check that the cluster nodes have an OpState of “Online” within the cluster domain, with command **lsrpnod**:

```
lead:~ # lsrpnod
Name OpState RSCTVersion
lead Online 2.5.0.2
lochnese Online 2.5.0.2
```

### ***Make a network tiebreaker resource***

Tiebreaker resource definition is not contained within any of the DB2-supplied TSA HADR scripts, so we add it here as a manual step. We use a network tiebreaker for HADR on Linux primarily because it is currently the only supported tiebreaker/quorum device in TSA Base Component for Linux, but also because it is highly appropriate for the tiebreaker to share the same point of failure that the HADR ports and/or remote client communications themselves are using. Detections of failures by the cluster through a network tiebreaker are not redundant or conflicting, and are truly indicative of the operational state of HADR and/or remote client connectivity, where the physical network connection to the tiebreaker is shared with the HADR and/or remote client connections.

Example 11-50 shows the **mkrsrc** command to make the IBM.TieBreaker class EXEC type resource. The single-line command syntax is split over three lines in our screen capture. The **chrsrc** command after that registers the resource as the quorum tiebreaker device for our peer domain cluster.

#### *Example 11-50 Making a network tiebreaker resource*

---

```
lead:~ # mkrsrc IBM.TieBreaker Type="EXEC" Name="leadloctie"
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=9.43.85.1
Log=1' PostReserveWaitTime=30;
lead:~ # chrsrc -c IBM.PeerNode OpQuorumTieBreaker="leadloctie"
```

---

According to the *TSA Base Component: Administrator's and User's Guide*, SC33-8272, we also need to explicitly refer to an external network address in a new file created by root in `/usr/sbin/cluster/netmon.cf` on both nodes. The functionality of this is complementary to the tiebreaker resource, specifically to detect network interface failures. It also avoids having two peer network interfaces being flagged offline by TSA even if only one network interface is failing to connect. Fortunately, the gateway IP address of the node's subnet is the recommended setting for this file, so we can use the same IP address as the network tiebreaker. Our *netmon.cf* files are created as shown in Example 11-51.

*Example 11-51 Additional required netmon.cf file*

---

```
lead:/usr/sbin # mkdir cluster
lead:/usr/sbin # cd cluster
lead:/usr/sbin/cluster # vi netmon.cf

lochnese:/usr/sbin # mkdir cluster
lochnese:/usr/sbin # cd cluster
lochnese:/usr/sbin/cluster # vi netmon.cf
```

---

*netmon.cf* for both our nodes contains the following two lines:

```
# this is default gateway for all interfaces in 9.43.86.x
network 9.43.85.1
```

### **Set up network equivalencies on each cluster node for TSA**

This allows TSA to attempt to keep a node available for DB2 communications even if one of its network interfaces fails. If all network interfaces fail, TSA detects this and takes appropriate action to make that node offline and switches the other node online. A network equivalency is required even if there is only one NIC per node, or a bonded definition exists to appear as a single NIC per node.

We use the **mkequ** command to perform this task as shown in Example 11-52. In our scenario, we have two physical NICs per cluster node, bonded to look like a single NIC to external connections, so our network equivalency is set up with only a single interface, just as if there were only one physical NIC. We use the “bond0” interface name, whereas an actual single NIC definition would probably be named “eth0”. The **lsequ** command is then used from either node to display and confirm the defined equivalencies.

*Example 11-52 Defining virtual network interface equivalency*

---

```
####Primary node:
lead:~ # mkequ -D "Name like 'bond0' & NodeNameList='lead' "
virpubnic_lead IBM.NetworkInterface

####Standby node:
lochnese:/ # mkequ -D "Name like 'bond0' & NodeNameList='lochnese' "
virpubnic_lochnese IBM.NetworkInterface

####Either node:
lochnese:/ # lsequ -A b
Displaying Equivalency information:
All Attributes

Equivalency 1:
```



```

Name = virpubnic_lochnese
MemberClass = IBM.NetworkInterface
Resource:Node[Membership] = {}
SelectString = "Name like 'bond0' &
NodeNameList='lochnese' "
SelectFromPolicy = ANY
MinimumNecessary = 1
Subscription = {}
ActivePeerDomain = hadr91
Resource:Node[ValidSelectResources] =
{bond0:lochnese.itsosj.sanjose.ibm.com}
Resource:Node[InvalidResources] = {}
ConfigValidity =
AutomationDetails[CompoundState] = Automation

```

#### Equivalency 2:

```

Name = virpubnic_lead
MemberClass = IBM.NetworkInterface
Resource:Node[Membership] = {}
SelectString = "Name like 'bond0' &
NodeNameList='lead' "
SelectFromPolicy = ANY
MinimumNecessary = 1
Subscription = {}
ActivePeerDomain = hadr91
Resource:Node[ValidSelectResources] =
{bond0:lead.itsosj.sanjose.ibm.com}
Resource:Node[InvalidResources] = {}
ConfigValidity =
AutomationDetails[CompoundState] = Automation

```

If you have more than one NIC and wish to make an equivalency containing multiple network adapters, Example 11-53 illustrates the syntax you might use to achieve this, for instance, with two network adapters defined as eth0 and eth1 on each node.

#### *Example 11-53 Making an equivalency for the cluster*

```

lead:~ # mkequ -D "(Name like 'eth0' | Name like 'eth1') & \
NodeNameList='lead'" virpubnic_lead IBM.NetworkInterface

lochnese:~ # mkequ -D "(Name like 'eth0' | Name like 'eth1') & \
NodeNameList='lochnese'" virpubnic_lochnese IBM.NetworkInterface

```

**Note:** In order for failover to work properly between multiple NICs on any given node in a TSA network equivalency, do not define more than one network interface on a node to the same subnet. Independent routes must exist in order to maintain communications to each remaining NIC even if one or more network interfaces fail.

This restriction does not apply to the single network adapter equivalency definition we use for our test environment.

### **Register virtual network interfaces and match DB2 instances to TSA**

After setting the TSA scripts to match requirements with the information given in “Using TSA with SSH:” on page 425, we arrive at the stage where we use the **regdb2sa1in** command. Expanding the command literally, “reg-db2-sa-lin” is designed to register DB2 with TSA in Linux. Among other functions, **regdb2sa1in** defines a virtual IP address to represent the network interface equivalencies created in the step “Set up network equivalencies on each cluster node for TSA” on page 430. The virtual IP definition requires use of an unused IP address inside the same subnet as the network interface equivalencies.

Example 11-54 sets out the steps where the root user issues the **regdb2sa1in** command on each node to register a virtual IP address against our IP equivalency resources, and to match the DB2 instances to TSA.

#### *Example 11-54 Registering DB2 and virtual network interface to TSA*

---

```
###Primary node:
lead:/software # cd /opt/ibm/db2/V9.1/ha/tsa/
lead:/opt/ibm/db2/V9.1/ha/tsa # ./regdb2sa1in -a db2inst0 -r -i 9.43.86.250 -n
255.255.252.0 -s
Distributing policy to node lead via ssh...
Warning: Permanently added the RSA host key for IP address '9.43.86.68' to the list of
known hosts.
Distributing policy to node lochnese via ssh...

About to register db2inst0 with TSA

Checking cluster validity...

Stopping instance db2inst0...

Checking resources able to host instance db2inst0 (partition 0)
  Checking lead ...

Making resource group db2_db2inst0_lead_0-rg ...
Making IP resource db2_db2inst0_lead_0-rs_ip ...
Making DB2 resource db2_db2inst0_lead_0-rs ...
Making relationship db2_db2inst0_lead_0-rg_IP_do...
```

```

Online resource group db2_db2inst0_lead_0-rg ...

db2_db2inst0_lead_0-rg is now online

Done, instance is now HA

###Standby node:
lochnese:/opt/ibm/db2/V9.1/ha/tsa # ./regdb2salin -a db2inst1 -r -i 9.43.86.251 -n
255.255.252.0 -s
Distributing policy to node lead via ssh...
Distributing policy to node lochnese via ssh...

About to register db2inst1 with TSA

Checking cluster validity...

Stopping instance db2inst1...

Checking resources able to host instance db2inst1 (partition 0)
  Checking lochnese ...

Making resource group db2_db2inst1_lochnese_0-rg ...
Making IP resource db2_db2inst1_lochnese_0-rs_ip ...
Making DB2 resource db2_db2inst1_lochnese_0-rs ...
Making relationship db2_db2inst1_lochnese_0-rg_IP_do...

Online resource group db2_db2inst1_lochnese_0-rg ...

db2_db2inst1_lochnese_0-rg is now online

Done, instance is now HA

```

---

It should be noted here that **regdb2salin** can take a few minutes to execute, with apparent long pauses in a couple of locations. If it has not completed after about five minutes, the script can be cancelled with <Ctrl>-<C>, cleaning up the definitions with the **unregdb2salin** command script, and restarting **regdb2salin** with the verbose flag to diagnose.

### ***Verify the configuration***

Confirm that we now have DB2 resource groups which can be displayed, stopped, and restarted with the following TSA commands:

- ▶ **lsrg**: List resource group
- ▶ **lssam**: List system automation
- ▶ **chrg**: Change status of resource group
- ▶ **getstatus**: Tabular status output of resources, groups, nodes
- ▶ **lsrel**:- List dependency relationships

**Note:** Now that DB2 instances have been registered under TSA control with the `regdb2salin` command, they should not be stopped or started with the DB2 standard `db2stop` or `db2start` commands, nor should they be controlled by the DB2 fault monitor or fault monitor coordinator daemon. The correct method of starting or stopping a DB2 instance is through the `chrg` command issued by the root user, which makes use of the `db2gcf` command behind the scenes, integrated with the cluster's resource group status and dependency management. In clustering nomenclature, a resource group request can bring a DB2 instance resource "offline" as an equivalent to `db2stop`, and request "online" as an equivalent to `db2start`.

The DB2 instance user IDs will not have required cluster permissions to issue the `chrg` commands, so they must be performed by the root user.

Native DB2 commands can be safely used for maintenance purposes, if the cluster automation is disabled. TSA automation of all resources can be disabled or individual resources can be excluded from TSA control with the `samctr1` command.

Example 11-55 shows these commands being executed on each server immediately after each `regdb2salin` was run, then finally, `getstatus` and `lsrel` commands showing a summary of these cluster object definitions and their dependency relationships for both nodes. At the time of running these commands on the first server, the second resource group had not been populated by `regdb2salin` yet, so it does not display in `lsrg` or `lsam`.

In practice, stopping (bringing *Offline*) or starting (bringing *Online*) resources such as a DB2 instance inside a resource group can take time, so repeated execution of the `lsam` command is required to see whether the desired action has occurred in the cluster. Some of these initial TSA definition commands are specific to the `/opt/ibm/db2/V9.1/ha/tsa` subdirectory and are executed with explicit path qualifiers (`./regdb2salin`), but others are already in the installed TSA path definition and can be executed with implicit path (`lsam`, for example).

*Example 11-55 Confirming resource group status, and stop/start*

---

```
lead:/opt/ibm/db2/V9.1/ha/tsa # lsrg
Resource Group names:
db2_db2inst0_lead_0-rg
lead:/opt/ibm/db2/V9.1/ha/tsa # lsam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
   |- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
   |- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
   |- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
lead:/opt/ibm/db2/V9.1/ha/tsa # chrg -o offline db2_db2inst0_lead_0-rg
lead:/opt/ibm/db2/V9.1/ha/tsa # lsam
```

```

Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Offline
|- Offline IBM.Application:db2_db2inst0_lead_0-rs
  '- Offline IBM.Application:db2_db2inst0_lead_0-rs:lead
'- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
  '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
lead:/opt/ibm/db2/V9.1/ha/tsa # chrg -o online db2_db2inst0_lead_0-rg
lead:/opt/ibm/db2/V9.1/ha/tsa # lssam
Pending online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst0_lead_0-rs
  '- Pending online IBM.Application:db2_db2inst0_lead_0-rs:lead
'- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
lead:/opt/ibm/db2/V9.1/ha/tsa # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
  '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
'- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead

```

**Tochnese:/opt/ibm/db2/V9.1/ha/tsa # lsrsg**

Resource Group names:

db2\_db2inst1\_lochnese\_0-rg

db2\_db2inst0\_lead\_0-rg

lochnese:/opt/ibm/db2/V9.1/ha/tsa # **lssam**

Online IBM.ResourceGroup:db2\_db2inst0\_lead\_0-rg Nominal=Online

|- Online IBM.Application:db2\_db2inst0\_lead\_0-rs

'- Online IBM.Application:db2\_db2inst0\_lead\_0-rs:lead

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip:lead

Online IBM.ResourceGroup:db2\_db2inst1\_lochnese\_0-rg Nominal=Online

|- Online IBM.Application:db2\_db2inst1\_lochnese\_0-rs

'- Online IBM.Application:db2\_db2inst1\_lochnese\_0-rs:lochnese

'- Online IBM.ServiceIP:db2\_db2inst1\_lochnese\_0-rs\_ip

'- Online IBM.ServiceIP:db2\_db2inst1\_lochnese\_0-rs\_ip:lochnese

lochnese:/opt/ibm/db2/V9.1/ha/tsa # **chrg -o offline db2\_db2inst1\_lochnese\_0-rg**

lochnese:/opt/ibm/db2/V9.1/ha/tsa # lssam

Online IBM.ResourceGroup:db2\_db2inst0\_lead\_0-rg Nominal=Online

|- Online IBM.Application:db2\_db2inst0\_lead\_0-rs

'- Online IBM.Application:db2\_db2inst0\_lead\_0-rs:lead

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip:lead

Pending *offline* IBM.ResourceGroup:db2\_db2inst1\_lochnese\_0-rg Nominal=*Offline*

|- Pending *offline* IBM.Application:db2\_db2inst1\_lochnese\_0-rs

'- Pending *offline* IBM.Application:db2\_db2inst1\_lochnese\_0-rs:lochnese

'- Online IBM.ServiceIP:db2\_db2inst1\_lochnese\_0-rs\_ip

'- Online IBM.ServiceIP:db2\_db2inst1\_lochnese\_0-rs\_ip:lochnese

lochnese:/opt/ibm/db2/V9.1/ha/tsa # **lssam**

Online IBM.ResourceGroup:db2\_db2inst0\_lead\_0-rg Nominal=Online

|- Online IBM.Application:db2\_db2inst0\_lead\_0-rs

'- Online IBM.Application:db2\_db2inst0\_lead\_0-rs:lead

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip

'- Online IBM.ServiceIP:db2\_db2inst0\_lead\_0-rs\_ip:lead

Offline IBM.ResourceGroup:db2\_db2inst1\_lochnese\_0-rg Nominal=*Offline*

```

|- Offline IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Offline IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
'- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
  '- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
lochnese:/opt/ibm/db2/V9.1/ha/tsa # chrg -o online db2_db2inst1_lochnese_0-rg
lochnese:/opt/ibm/db2/V9.1/ha/tsa # Issam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
  '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Pending online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Pending online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
lochnese:/opt/ibm/db2/V9.1/ha/tsa # Issam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
  '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese

```

```
lead:/opt/ibm/db2/V9.1/ha/tsa # ./getstatus
```

```
-- Resource Groups and Resources --
```

| Group Name                 | Resources                     |
|----------------------------|-------------------------------|
| -----                      | -----                         |
| db2_db2inst0_lead_0-rg     | db2_db2inst0_lead_0-rs        |
| db2_db2inst0_lead_0-rg     | db2_db2inst0_lead_0-rs_ip     |
| -                          | -                             |
| db2_db2inst1_lochnese_0-rg | db2_db2inst1_lochnese_0-rs    |
| db2_db2inst1_lochnese_0-rg | db2_db2inst1_lochnese_0-rs_ip |
| -                          | -                             |

```
-- Resources --
```

| Resource Name              | Node Name | State  |
|----------------------------|-----------|--------|
| -----                      | -----     | ----   |
| db2_db2inst0_lead_0-rs     | lead      | Online |
| -                          | -         | -      |
| db2_db2inst0_lead_0-rs_ip  | lead      | Online |
| -                          | -         | -      |
| db2_db2inst1_lochnese_0-rs | lochnese  | Online |
| -                          | -         | -      |

```

db2_db2inst1_lochnese_0-rs_ip      lochnese      Online
-
lead:/opt/ibm/db2/V9.1/ha/tsa # 1sre1
Displaying Managed Relations :

Name                               Class:Resource:Node[Source]      ResourceGroup[Source]
db2_db2inst1_lochnese_0-rg_IP_do  IBM.Application:db2_db2inst1_lochnese_0-rs
db2_db2inst1_lochnese_0-rg
db2_db2inst0_lead_0-rg_IP_do      IBM.Application:db2_db2inst0_lead_0-rs  db2_db2inst0_lead_0-rg

```

---

### **Adding DB2 HADR resources to a third TSA resource group**

At this stage, we have two DB2 instance resources defined to TSA in unrelated resource groups on separate cluster nodes. We now need to run an additional registration command specifically for HADR, which adds a third resource group containing the HADR database pair.

We run the command **reghadrsa1 in** against a Connected, Peer state HADR database pair. After the **regdb2sa1 in** command was issued on the standby server, DB2 will have been recycled, but the HADR standby database may not have been reactivated.

To correct this, we su to the DB2 instance on the standby node, and run:

```
db2 activate db <dbname>
```

The state of the HADR database pair can be checked with the following command:

```
db2pd -d <dbname> -hadr
```

We expect to see the databases as connected and in Peer state before proceeding.

The next step is to define our Connected, Peer state HADR database pair into the TSA resource group as primary and standby HADR resources, using the **reghadrsa1 in** command executed by the root user only on the HADR primary node, shown in Example 11-56.

#### *Example 11-56 Defining HADR databases into a third TSA resource group*

```
lead:/opt/ibm/db2/V9.1/ha/tsa # ./reghadrsa1in -a db2inst0 -b db2inst1
-d test
```

```

About to register db2hadr_test with TSA ...
Creating HADR resource group ...
Making resource group db2hadr_test-rg ...
Making resource db2hadr_test-rs ...
Online resource group db2hadr_test-rg ...

```

Done, HADR is now registered with the TSA framework

---

The **getstatus** and **lssam** commands confirm that we now have a third resource group as shown in Example 11-57.

*Example 11-57 Verify the definition*

```

lead:/opt/ibm/db2/V9.1/ha/tsa # ./getstatus
-- Resource Groups and Resources --

                Group Name                Resources
                -----                -
db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs
db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs_ip
-
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs_ip
-
                db2hadr_test-rg                db2hadr_test-rs
                -
-- Resources --

                Resource Name                Node Name                State
                -----                -
db2_db2inst0_lead_0-rs                lead                Online
-
db2_db2inst0_lead_0-rs_ip                lead                Online
-
db2_db2inst1_lochnese_0-rs                lochnese                Online
-
db2_db2inst1_lochnese_0-rs_ip                lochnese                Online
-
                db2hadr_test-rs                lead                Online
                db2hadr_test-rs                lochnese                Offline
                -

lead:/opt/ibm/db2/V9.1/ha/tsa # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
  '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
|- Online IBM.Application:db2hadr_test-rs

```



```
|- Online IBM.Application:db2hadr_test-rs:lead
'- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

The `getstatus` and `lssam` command output can show operational state values of `Online`, `Offline`, and `Unknown` for DB2 HADR resources. These meanings are subtly different for HADR than the normal interpretation, as a working HADR standby database is never offline in the true sense of the word. Instead, the following meanings are ascribed:

- ▶ **Online:** The database is active and in HADR primary role.
- ▶ **Offline:** The database is active and in HADR standby role.
- ▶ **Unknown:** The database is either not in HADR Peer state, or is in standard role (not in an HADR role), or the database is deactivated and not otherwise accessible via HADR connectivity methods.

## Bringing an online TSA HADR cluster offline

Generally speaking, if the entire cluster domain is to be brought down for maintenance or cluster re-definition, the root user will first need to issue successive `chrg -o offline <resource group>` commands. The HADR resource group should be brought *Offline* first, then each of the resource groups for the two nodes, as shown in Example 11-58.

*Example 11-58 the stages of bringing a cluster offline correctly*

---

```
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

lead:~ # chrg -o offline db2hadr_test-rg
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
```

```

Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Offline
  '- Offline IBM.Application:db2hadr_test-rs
    |- Offline IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lead:~ # chrg -o offline db2_db2inst0_lead_0-rg
lead:~ # lssam
Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Offline
|- Offline IBM.Application:db2_db2inst0_lead_0-rs
  '- Offline IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Offline
  '- Offline IBM.Application:db2hadr_test-rs
    |- Offline IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lead:~ # chrg -o offline db2_db2inst1_lochnese_0-rg
lead:~ # lssam
Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Offline
|- Offline IBM.Application:db2_db2inst0_lead_0-rs
  '- Offline IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Offline IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Offline
|- Offline IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Offline IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Offline
  '- Offline IBM.Application:db2hadr_test-rs
    |- Offline IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

Bringing resource groups offline out of order will result in a hanging *Pending Offline* status on at least one of the non HADR resource-groups, while the HADR resource group lies in an Unknown/Offline status, illustrated in Example 11-59, where **chrg -o offline** was executed against db2\_db2inst0\_lead\_0-rg, then db2\_db2inst1\_lochnese\_0-rg, then db2hadr\_test-rg. This is because when the resource group for the underlying DB2 instance is offline, the status for one of the

HADR resources cannot be resolved out of Unknown operational state into Offline operational state. This also means that after bringing all the resource groups back online, manual activation of each HADR database in the pair will need to be done by the DB2 instance user, issuing a **db2 activate db <dbname>** command.

*Example 11-59 Cluster status after bringing resource groups offline out of order*

---

```

lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Offline IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Offline
  |- Offline IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Offline IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Pending offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Offline
  '- Unknown IBM.Application:db2hadr_test-rs
    |- Unknown IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

If the cluster is to be redefined, the offline resources and resource groups can be removed with the **rmrgmbr** and **rmrg** commands, as shown in Example 11-60, with the **lssam** command showing the state of the cluster definition.

*Example 11-60 Removing resource groups and members*

---

```

lead:~ # rmrgmbr -g db2hadr_test-rg
lead:~ # rmrgmbr -g db2_db2inst1_lochnese_0-rg
lead:~ # rmrgmbr -g db2_db2inst0_lead_0-rg
lead:~ # lssam
Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Offline
Offline IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Offline
Offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Offline
lead:~ # rmrg db2hadr_test-rg
lead:~ # rmrg db2_db2inst1_lochnese_0-rg
lead:~ # rmrg db2_db2inst0_lead_0-rg
lead:~ # lssam
lssam: No resource groups defined or cluster is offline!

```

---

Note that we do not need to remove or stop the peer domain definition in order to perform maintenance/redefinition of cluster resources and resource groups.

For these and other cluster definition and maintenance commands, refer to the *RSCT Administration Guide*. SA22-7889.

## 11.4.2 Testing topology response to common failures

In this section we go through the steps of a standard set of tests to prove that the TSA controlled HADR cluster is capable of handling common failure scenarios in an appropriate manner.

Now that we have an online cluster of an HADR database pair managed by TSA in resource groups, we can test unforced, forced, and automated takeovers.

### Prerequisites

All tests assume as a prerequisite that our HADR database pair TEST is active, connected, and in Peer state, and that our HADR primary role is held by lead, and HADR standby is lochnese, as shown in Example 11-61.

*Example 11-61 The assumed starting point for all our testing scenarios*

```
db2inst0@lead:~> lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
db2inst0@lead:~> db2pd -d test -hadr

Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:53:09

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary Peer                               Sync      0                    0

ConnectStatus ConnectTime                               Timeout
Connected      Wed Apr 16 16:26:02 2008 (1208388362) 15

LocalHost                               LocalService
lead                                       55101
```

| RemoteHost   | RemoteService | RemoteInstance     |
|--------------|---------------|--------------------|
| lochnese     | 55102         | db2inst1           |
| PrimaryFile  | PrimaryPg     | PrimaryLSN         |
| S0000002.LOG | 0             | 0x00000000017A0000 |
| StandByFile  | StandByPg     | StandByLSN         |
| S0000002.LOG | 0             | 0x00000000017A0000 |

---

## Monitoring scripts

During testing of certain actions or failures, it is useful to monitor the status of the cluster. We use a single line shell script which repeatedly issues an `lssam` command, echoes an empty line, and sleeps for 5 seconds:

```
while : ;do lssam;echo " ";sleep 5;done
```

We also monitor the DB2 connectivity of the HADR database pair with a simple looping script/batch file. Our example runs from a DB2 client on Microsoft Windows, so it uses the following syntax:

```
:START
db2 connect to testlead user db2inst0 using passblah
db2 connect reset
goto START
```

This batch file assumes the following catalog TCP/IP node and database alias definition executed on the remote client at the DB2 command line:

```
db2 catalog tcpip node testlead remote 9.43.86.250 server 55000
db2 catalog db test as testlead at node testlead
```

We also perform a catalog TCP/IP node and database for the alternate server, just to manually test Automatic Client Reroute (ACR) functionality:

```
db2 catalog tcpip node testloch remote 9.43.86.250 server 55000
db2 catalog db test as testloch at node testloch
```

In order to avoid causing connection failures when using an ACR alternate server definition, the IP addresses of the servers we are using should be added to the remote client's `/etc/hosts` file (In Microsoft Windows, this is generally the `%SystemRoot%\system32\drivers\etc\hosts` file). As with the primary TCP/IP node IP address, we use the virtual/equivalency IP address for both cluster nodes in the client's `/etc/hosts`, rather than the IP address of the physical port:

```
llead 9.43.86.250
lochnese 9.43.86.251
```

This is due to the level of abstraction with which TSA has registered DB2 HADR resources. The purpose is so TSA can switch the virtual IP addresses with each other to always keep the database currently with the HADR primary role on the same address. If the virtual IP address being used by the TSA definition is likely to frequently change, or there are too many clients to manage `/etc/hosts` entries for, a more permanent and dynamic solution would require a DNS entry and an alternate host name to be used by DB2 clients. ACR definitions on the server would also need to use each alternate host name, as these are used by the remote clients and need to be resolved correctly to the virtual IP address rather than the physical IP address.

## Failover testing

We perform seven types of tests for our cluster environment, and monitor the results:

- ▶ Unforced takeover
- ▶ DB2 instance failure on the primary node
- ▶ DB2 instance failure on the standby node
- ▶ Planned resource group downtime - primary DB2 Instance
- ▶ Planned resource group downtime - standby DB2 Instance
- ▶ Test of highly available network adapters
- ▶ Testing failure of a cluster node

These tests are merely representative of the myriad of different ways in which a cluster can go offline in either a planned or unplanned manner. Our example is more basic than many configurations which are possible, specifically with multiple public and private network topologies, which have been explored with the AIX testing scenarios (our AIX testing hardware has more network adapters than the Linux hardware).

### ***Unforced takeover***

This would normally be achieved by a `db2 takeover hadr for db <dbname>` command issued against the DB2 instance which wanted to take on the HADR primary role.

Under TSA control, this action must instead be performed with the `rgreq` (resource group request) command by the root user. Example 11-62 shows the initial status of node lead with the “online” primary role.

#### *Example 11-62 Initial status*

---

```
lead:/opt/ibm/db2/V9.1/ha/tsa # ./getstatus
```

```
-- Resource Groups and Resources --
```

| Group Name | Resources |
|------------|-----------|
| -----      | -----     |

```

db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs
db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs_ip
-
-
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs_ip
-
-
db2hadr_test-rg db2hadr_test-rs
-
-
-- Resources --

Resource Name          Node Name          State
-----
db2_db2inst0_lead_0-rs lead              Online
-
-
db2_db2inst0_lead_0-rs_ip lead              Online
-
-
db2_db2inst1_lochnese_0-rs lochnese          Online
-
-
db2_db2inst1_lochnese_0-rs_ip lochnese          Online
-
-
db2hadr_test-rs lead              Online
db2hadr_test-rs lochnese          Offline
-
-

```

---

We perform the **rgreq** command with the root user to switch the role manually. The **getstatus** output shows node lochnese as the “online” HADR primary role. See Example 11-63.

*Example 11-63 Unforced takeover to assign HADR primary role*

```

lead:/opt/ibm/db2/V9.1/ha/tsa # rgreq -o move -n lead db2hadr_test-rg
Action on resource group "db2hadr_test-rg" returned Token
"0x28affe7fa1843530d58e0648c27f0800" .

```

```

lead:/opt/ibm/db2/V9.1/ha/tsa # ./getstatus

```

```

-- Resource Groups and Resources --

```

```

-----
Group Name          Resources
-----
db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs
db2_db2inst0_lead_0-rg db2_db2inst0_lead_0-rs_ip
-
-
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs
db2_db2inst1_lochnese_0-rg db2_db2inst1_lochnese_0-rs_ip

```

```

                -                -
                db2hadr_test-rg    db2hadr_test-rs
                -                -

-- Resources --

                Resource Name      Node Name      State
                -----            -
                db2_db2inst0_lead_0-rs      lead      Online
                -                -
                db2_db2inst0_lead_0-rs_ip    lead      Online
                -                -
                db2_db2inst1_lochnese_0-rs    lochnese   Online
                -                -
                db2_db2inst1_lochnese_0-rs_ip  lochnese   Online
                -                -
                db2hadr_test-rs      lead      Offline
                db2hadr_test-rs      lochnese   Online
                -                -

```

---

The **db2pd -d test -hadr** output by the DB2 instance on lochnese confirms this result. See Example 11-64.

*Example 11-64 db2pd output*

---

```

db2inst1@lochnese: /> db2pd -d test -hadr

Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:19:27

HADR Information:
Role      State      SyncMode HeartBeatsMissed LogGapRunAvg (bytes)
Primary Peer      Sync      0              0

ConnectStatus ConnectTime      Timeout
Connected      Wed Apr 16 16:24:41 2008 (1208388281) 15

LocalHost      LocalService
lochnese      55102

RemoteHost      RemoteService      RemoteInstance
lead      55101      db2inst0

PrimaryFile PrimaryPg PrimaryLSN
S0000002.LOG 0      0x00000000017A0000

StandByFile StandByPg StandByLSN
S0000002.LOG 0      0x00000000017A0000

```

---



We can put the HADR primary role back on lead with the same command, but with `lochnese` as the node name parameter. This time, we monitor the status of the HADR resource group using the `lssam` command. See Example 11-65.

*Example 11-65 Resume to the initial role of the HADR pair*

---

```
lead:/opt/ibm/db2/V9.1/ha/tsa # rgreg -o move -n lochnese db2hadr_test-rg
Action on resource group "db2hadr_test-rg" returned Token
"0x28affe7fa1843530f7950648afdd0400" .

lead:/opt/ibm/db2/V9.1/ha/tsa # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Pending online IBM.ResourceGroup:db2hadr_test-rg Request=Move Nominal=Online
  '- Pending online IBM.Application:db2hadr_test-rs
    |- Pending online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lead:/opt/ibm/db2/V9.1/ha/tsa # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

***DB2 instance failure on the primary node***

The easiest way to achieve this effect is to simply terminate the primary DB2 instance with the `db2_kill` command:

```
db2inst0@lead:~/script> db2_kill
ipclean: Removing DB2 engine and client's IPC resources for db2inst0.
```

Crossing over to the standby node, we issue our looping `Issam` script to monitor the status of the resources. Initially, in Example 11-66, we see that TSA has not yet detected anything - all resources are listed in their nominal state.

*Example 11-66 Output of Issam immediately after db2\_kill*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

After several seconds, TSA detects that the primary DB2 is stopped and sets those DB2 instance resources to *Pending Online* state, as seen in Example 11-67.

*Example 11-67 Output of Issam after detecting DB2 instance is down*

---

```
Pending online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst0_lead_0-rs
    '- Pending online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

The first action preferred by TSA is to simply attempt to restart the DB2 resource on the HADR Primary node (rather than switching HADR roles). Sure enough, after a few more seconds, all resources are back to nominal states, with DB2 Online again on the primary node. The output from a `db2pd` command shows HADR is also back in Connected, Peer state, with the HADR Primary role assigned to the original primary node in Example 11-68.

*Example 11-68 Output of lssam, db2pd showing all is back to normal*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
  '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
|- Online IBM.Application:db2hadr_test-rs
  |- Online IBM.Application:db2hadr_test-rs:lead
  '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

**db2inst0@lead:~/script> db2pd -d test -hadr**

Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:13:20

HADR Information:

| Role         | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|--------------|-------|----------|------------------|----------------------|
| Primary Peer |       | Sync     | 0                | 0                    |

| ConnectStatus | ConnectTime                           | Timeout |
|---------------|---------------------------------------|---------|
| Connected     | Wed Apr 16 18:04:54 2008 (1208394294) | 15      |

| LocalHost | LocalService |
|-----------|--------------|
| lead      | 55101        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| lochnese   | 55102         | db2inst1       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000002.LOG | 0         | 0x00000000017A0000 |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000002.LOG | 0         | 0x00000000017A0000 |

---

***DB2 instance failure on the standby node***

As with the example of terminating the DB2 instance on the primary node, we achieve this with an execution of the `db2_kill` command issued by the DB2 instance user on the standby node.

```
db2inst1@lochnese:~> db2_kill
ipclean: Removing DB2 engine and client's IPC resources for db2inst1.
```

The **db2pd** output in Example 11-69 confirms that at least **db2pd** is no longer able to attach to the instance, with the obvious implication that nothing else would be able to attach either.

*Example 11-69 Output of db2pd after db2\_kill on HADR Standby node*

---

```
db2inst1@lochnese:~> db2pd -d test -hadr
Unable to attach to database manager on partition 0.
Please ensure the following are true:
  - db2start has been run for the partition.
  - db2pd is being run on the same physical machine as the partition.
  - DB2NODE environment variable setting is correct for the partition
    or db2pd -dbp setting is correct for the partition.
```

---

Immediately issuing our looping **Issam** script in Example 11-70 shows TSA not yet having registered anything amiss — all resources are still in nominal states.

*Example 11-70 Output of Issam immediately after db2\_kill*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

After several seconds, we see in Example 11-71 that the fault is detected as a *Pending Online* flag, and the first preference of TSA is to attempt a restart of that DB2 resource.

*Example 11-71 Output of Issam several seconds after db2\_kill*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
```

```

'- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Pending online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Pending online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
      '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

Another several seconds later, TSA has succeeded, and the DB2 instance is back Online in Example 11-72.

*Example 11-72 Output of lssam when TSA has brought DB2 instance back Online*

```

Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
      '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
        '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
      '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

We confirm that the DB2 HADR database pair are in Online : Offline (Primary : Standby) Connected Peer state with another **db2pd** command in Example 11-73, which means that TSA also successfully activated the DB2 HADR Standby database and it has reconnected with the HADR Primary.

*Example 11-73 Output of db2pd after TSA has brought DB2 instance back online*

```
db2inst1@lochnese:~> db2pd -d test -hadr
```

```
Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:02:03
```

```

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg
(bytes)
Standby Peer              Sync      0                0

ConnectStatus ConnectTime              Timeout
Connected   Thu Apr 17 18:01:38 2008 (1208480498) 15

LocalHost              LocalService
lochnese              55102

RemoteHost              RemoteService
RemoteInstance
lead                  55101              db2inst0

PrimaryFile PrimaryPg PrimaryLSN
S0000004.LOG 93          0x0000000001FFDF83

StandByFile StandByPg StandByLSN
S0000004.LOG 93          0x0000000001FFDF83

```

---

You might note that TSA is not reacting immediately, and that corrective action does take some time; how much time depends on a number of factors, and it differs depending on the environment it has been configured in. There are very few, (if any) overriding factors that can positively influence the performance of TSA in this respect. As a rough guide, no more than a couple of minutes should pass between a DB2 instance being stopped, and the appropriate corrective action being successfully completed to either bring it back online, or to failover to the DB2 instance on the other node if appropriate.

### ***Planned resource group downtime — primary DB2 instance***

While this can hardly be described as a failure in the true sense of the word, we must test that TSA can successfully bring down the resource group for the primary DB2 instance, and then bring it back up again.

As a root user, we issue a **chrg** (change resource group) command:

```
lead:~ # chrg -o offline db2_db2inst0_lead_0-rg
```

This command can be issued on either node. The **lssam** output shown in Example 11-74 confirms that the resource group state is *Offline*, and the HADR Primary database db2hadr\_test-rs:lead resource state has been set to *Unknown*.

*Example 11-74 The primary DB2 node's resource group is down*

---

```
lead:~ # lssam
Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Offline
  |- Offline IBM.Application:db2_db2inst0_lead_0-rs
    '- Offline IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Unknown IBM.Application:db2hadr_test-rs
    |- Unknown IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

The **chrg** command is issued to bring the resource group back Online. Successive **lssam** commands show the resource in Pending Online, then Online, and the HADR Primary resource (db2hadr\_test-rs:lead) changes from Unknown state back to Online.

Example 11-75 shows the db2\_db2inst0\_lead\_0-rg resource group as *Pending Online*, and the HADR database db2hadr\_test-rs resource as *Unknown*.

*Example 11-75 Output of chrg online and first lssam output*

---

```
lead:~ # chrg -o online db2_db2inst0_lead_0-rg
lead:~ # lssam
Pending online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst0_lead_0-rs
    '- Pending online
IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
```

```

'- Unknown IBM.Application:db2hadr_test-rs
  |- Unknown IBM.Application:db2hadr_test-rs:lead
  '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

Example 11-76 shows the db2\_db2inst0\_lead\_0-rg resource group as *Online*, however the HADR database db2hadr\_test-rs resource is still *Unknown*.

*Example 11-76 Second lssam output after chrg online*

---

```

lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Unknown IBM.Application:db2hadr_test-rs
    |- Unknown IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

In Example 11-77, we see that the HADR database resource db2hadr\_test-rs is also now *Online*, with each subordinate database in the Nominal Online and Offline state.

*Example 11-77 Final lssam output after chrg online*

---

```

db2inst0@lead:~> lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip

```



```

'- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

This final desired state is confirmed with the DB2 instance by issuing a **db2pd** command, as shown in Example 11-78, showing the HADR databases in Connected, Peer state.

*Example 11-78 The primary DB2 node's resource group is confirmed as Online.*

---

```
db2inst0@lead:~> db2pd -d test -hadr
```

```
Database Partition 0 -- Database TEST -- Active -- Up 0 days 15:25:55
```

```
HADR Information:
```

| Role    | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer  | Sync     | 0                | 0                    |

| ConnectStatus | ConnectTime                           | Timeout |
|---------------|---------------------------------------|---------|
| Connected     | Thu Apr 17 18:42:10 2008 (1208482930) | 15      |

| LocalHost | LocalService |
|-----------|--------------|
| lead      | 55101        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| lochnese   | 55102         | db2inst1       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000005.LOG | 219       | 0x000000000247B481 |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000005.LOG | 219       | 0x000000000247B481 |

---

### ***Planned resource group downtime — standby DB2 instance***

As with the prior example of bringing the primary resource group down and back up in a controlled manner, we now test that the same can be done with the standby resource group.

As the root user, we issue a **chrg** command against the standby resource group:

```
lead:~ # chrg -o offline db2_db2inst1_lochnese_0-rg
```

Example 11-79 shows the **lssam** output confirming that the resource group status is Offline.

*Example 11-79 Change resource group offline for the standby DB2 node.*

---

```
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Pending offline IBM.ResourceGroup:db2_db2inst1_lochinese_0-rg Nominal=Offline
  |- Pending offline IBM.Application:db2_db2inst1_lochinese_0-rs
    '- Pending offline
IBM.Application:db2_db2inst1_lochinese_0-rs:lochinese
  '- Online IBM.ServiceIP:db2_db2inst1_lochinese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochinese_0-rs_ip:lochinese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochinese

lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Offline IBM.ResourceGroup:db2_db2inst1_lochinese_0-rg Nominal=Offline
  |- Offline IBM.Application:db2_db2inst1_lochinese_0-rs
    '- Offline IBM.Application:db2_db2inst1_lochinese_0-rs:lochinese
  '- Offline IBM.ServiceIP:db2_db2inst1_lochinese_0-rs_ip
    '- Offline IBM.ServiceIP:db2_db2inst1_lochinese_0-rs_ip:lochinese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochinese
```

---

An interesting point to note here is that rather than changing to Unknown, the state of the actual HADR database resource remains as Offline to the Cluster. The actual state of the standby HADR database is definitely Offline, as confirmed with a **db2pd** command issued from both the HADR primary DB2 instance ID, and HADR standby DB2 instance ID. See Example 11-80.

*Example 11-80 Checking whether the HADR Standby database is Offline*

---

```
db2inst0@lead:~> db2pd -d test -hadr
```

```
Database Partition 0 -- Database TEST -- Active -- Up 0 days 15:44:57
```

```
HADR Information:
```

| Role    | State        | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|--------------|----------|------------------|----------------------|
| Primary | Disconnected | Sync     | 0                | 0                    |

```

ConnectStatus ConnectTime                               Timeout
Disconnected  Fri Apr 18 10:23:33 2008 (1208539413) 15

LocalHost                               LocalService
lead                                     55101

RemoteHost                               RemoteService   RemoteInstance
lochnese                                 55102           db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000005.LOG 219      0x000000000247B481

StandByFile StandByPg StandByLSN
S0000005.LOG 219      0x000000000247B481

```

```

db2inst1@lochnese:~> db2pd -d test -hadr
Unable to attach to database manager on partition 0.
Please ensure the following are true:
- db2start has been run for the partition.
- db2pd is being run on the same physical machine as the partition.
- DB2NODE environment variable setting is correct for the partition
  or db2pd -dbp setting is correct for the partition.

```

---

Example 11-81 shows the standby node being brought back Online with another **chrg** command issued by root, followed by **lssam** output. In our experience, the progression of the standby resource group's state back to Online did take several seconds longer than for the primary, and the **lssam** output from the root user showed a "Request=Lock" highlighted, whereas **lssam** output from root or DB2 instance IDs on the other nodes did not show the "Request=Lock". This highlighted text eventually also disappeared on the **lssam** output from the issuing root user.

---

*Example 11-81 Bringing the standby node back online*

---

```

lead:~ # chrg -o online db2_db2inst1_lochnese_0-rg
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
      '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
        '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Pending online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Pending online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese

```

```

Online IBM.ResourceGroup:db2hadr_test-rg Request=Lock Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Request=Lock Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

In Example 11-82, the output from **db2pd** shows an expected HADR Connected, Peer state.

*Example 11-82 An expected result after standby cluster node is brought back online*

---

```
db2inst0@lead:~> db2pd -d test -hadr
```

```
Database Partition 0 -- Database TEST -- Active -- Up 0 days 15:58:25
```

```
HADR Information:
```

| Role    | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer  | Sync     | 0                | 0                    |

```
ConnectStatus ConnectTime
```

```
Timeout
```

Connected Fri Apr 18 10:32:04 2008 (1208539924) 15

|              |           |                    |                |
|--------------|-----------|--------------------|----------------|
| LocalHost    |           | LocalService       |                |
| Lead         |           | 55101              |                |
| RemoteHost   |           | RemoteService      | RemoteInstance |
| lochinese    |           | 55102              | db2inst1       |
| PrimaryFile  | PrimaryPg | PrimaryLSN         |                |
| S0000005.LOG | 219       | 0x000000000247B481 |                |
| StandByFile  | StandByPg | StandByLSN         |                |
| S0000005.LOG | 219       | 0x000000000247B481 |                |

---

### ***Test of highly available network adapters***

This test is only relevant for architectures where multiple physical network interface cards (NICs) have been installed and either configured into the cluster definition, or bonded outside the control of the cluster to provide an equivalent high availability network. Our architecture uses two physical NICs on each server, which are mapped as slaves to a bonded single network adapter, which is in turn registered into a cluster equivalency definition with a unique virtual IP address on each server. Our bonded network configuration is explained earlier in “Network configuration” on page 416.

The purpose of this particular test is simply to confirm that the cluster can handle failing over communications from one NIC to another. This can be achieved locally by physically pulling out the primary ethernet cable from the back of the server, or remotely if using an unbonded pair by disabling the ethernet card definition in Linux with the **ifdown** command. Using **ifdown** on a bonded network adapter definition will corrupt the definition and functionality of the network as a whole.

What we would expect to see in a multiple NIC equivalency definition with at least one remaining active NIC, is that the node is kept online, with only a short period where the virtual IP address is being switched over from the primary NIC to the next available NIC in the equivalency group. In our bonded network definition, this switching is performed outside the cluster’s control, and in practice, the switching lag is not noticeable. In either case, where there is no available NIC remaining, we would expect to see the cluster take appropriate actions to set the resource group for that cluster node completely offline, as it does in “Testing failure of a cluster node” on page 465.

The following series of examples illustrate our testing in detail. First, it is helpful to see which part of the cluster we are attempting to test. In this case, we are dealing with the network equivalency cluster definitions. If our environment had used IBM.NetworkInterface equivalency definitions on a normal unbonded pair of

NIC definitions, which many sites will choose, it would appear similar to Example 11-83.

*Example 11-83 Listing equivalencies in an unbonded dual-NIC environment*

---

```
lochnese:/opt/ibm/db2/V9.1/ha/tsa # lsequ -A b
Displaying Equivalency information:
All Attributes

Equivalency 1:
  Name = virpubnic_lochnese
  MemberClass = IBM.NetworkInterface
  Resource:Node[Membership] = {}
  SelectString = "(Name like 'eth0' | Name like 'eth1') &
NodeNameList='lochnese'"
  SelectFromPolicy = ANY
  MinimumNecessary = 1
  Subscription = {}
  ActivePeerDomain = hadr91
  Resource:Node[ValidSelectResources] =
{eth0:lochnese.itsosj.sanjose.ibm.com,eth1:lochnese.itsosj.sanjose.ibm.com}
  Resource:Node[InvalidResources] = {}
  ConfigValidity =
AutomationDetails[CompoundState] = Automation

Equivalency 2:
  Name = virpubnic_lead
  MemberClass = IBM.NetworkInterface
  Resource:Node[Membership] = {}
  SelectString = "(Name like 'eth0' | Name like 'eth1') &
NodeNameList='lead'"
  SelectFromPolicy = ANY
  MinimumNecessary = 1
  Subscription = {}
  ActivePeerDomain = hadr91
  Resource:Node[ValidSelectResources] =
{eth0:lead.itsosj.sanjose.ibm.com,eth1:lead.itsosj.sanjose.ibm.com}
  Resource:Node[InvalidResources] = {}
  ConfigValidity =
AutomationDetails[CompoundState] = Automation
```

---

Instead, we defined our pair of NICs into a bonded network definition and our resulting `IBM.NetworkInterface` equivalency definitions are shown in Example 11-84. The only difference between equivalency definitions on unbonded and bonded network adapters, is that the *SelectString* and *ValidSelectResources* values have to include a pattern string matching all member network adapters. Those values for multiple unbonded network adapters will have each individual “eth0” and “eth1” adapter listed, whereas our bonded values only have the single “bond0” adapter, as specified in Example 11-52 on page 430.

*Example 11-84 Listing equivalencies in an bonded dual-NIC environment*

---

```
lead:~ # lsequ -A b
Displaying Equivalency information:
All Attributes

Equivalency 1:
  Name = virpubnic_lochnese
  MemberClass = IBM.NetworkInterface
  Resource:Node[Membership] = {}
  SelectString = "Name like 'bond0' &
NodeNameList='lochnese' "
  SelectFromPolicy = ANY
  MinimumNecessary = 1
  Subscription = {}
  ActivePeerDomain = hadr91
  Resource:Node[ValidSelectResources] = {bond0:lochnese.itsosj.sanjose.ibm.com}
  Resource:Node[InvalidResources] = {}
  ConfigValidity =
  AutomationDetails[CompoundState] = Automation

Equivalency 2:
  Name = virpubnic_lead
  MemberClass = IBM.NetworkInterface
  Resource:Node[Membership] = {}
  SelectString = "Name like 'bond0' & NodeNameList='lead' "
  SelectFromPolicy = ANY
  MinimumNecessary = 1
  Subscription = {}
  ActivePeerDomain = hadr91
  Resource:Node[ValidSelectResources] = {bond0:lead.itsosj.sanjose.ibm.com}
  Resource:Node[InvalidResources] = {}
  ConfigValidity =
  AutomationDetails[CompoundState] = Automation
```

---

As you can see, the latter bonded network definition appears to the cluster as only a single network adapter, and as such, will resemble simple cluster implementations which only have a single adapter connecting the two DB2 server cluster nodes. In this configuration, neither adapter can be considered “primary”, so it makes little difference which network cable is removed. We tested removing one, then replacing it and removing the other. While our bonded example allows for high availability (and load balancing among other bonded network features), it is completely outside the control or visibility of the cluster. The former definition relies on the cluster in order to failover from one network adapter to the other.

To capture the cluster activity during our test, we first commence execution of the looping `lssam` monitor script on the standby node. Example 11-85 shows the initial output — nothing has been detected by TSA, all resources are in nominal states.

*Example 11-85 Initial output of lssam - all resources in nominal state*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

We also issue a ping command to the primary node's IP address from the standby node, shown in Example 11-86. This could also be done from a remote client as an equally valid test. Initially, all is well, ping response is good as expected.

*Example 11-86 Output of ping command immediately before cable disconnect*

---

```
lochnese:~ # ping 9.43.86.84
PING 9.43.86.84 (9.43.86.84) 56(84) bytes of data.
64 bytes from 9.43.86.84: icmp_seq=1 ttl=64 time=0.014 ms
64 bytes from 9.43.86.84: icmp_seq=2 ttl=64 time=0.014 ms
64 bytes from 9.43.86.84: icmp_seq=3 ttl=64 time=0.022 ms
...
```

---

We perform our test by physically removing a network cable from the back of the primary cluster node. It is impossible to tell exactly where in the ping output the disconnect occurs - as we can see in Example 11-87, there are no time-outs, and the *time=x.xxx ms* values for all our ping results are within acceptable ranges of 0.007 and 0.023 milliseconds. There is also no apparent cut-over point in these values indicating that available network bandwidth has dropped, as that range of values goes up and down even while both network cables are connected.

*Example 11-87 Output of ping command before and after cable disconnect*

---

```
64 bytes from 9.43.86.84: icmp_seq=10 ttl=64 time=0.008 ms
64 bytes from 9.43.86.84: icmp_seq=11 ttl=64 time=0.013 ms
64 bytes from 9.43.86.84: icmp_seq=12 ttl=64 time=0.009 ms
...
64 bytes from 9.43.86.84: icmp_seq=27 ttl=64 time=0.021 ms
64 bytes from 9.43.86.84: icmp_seq=28 ttl=64 time=0.023 ms
64 bytes from 9.43.86.84: icmp_seq=29 ttl=64 time=0.021 ms
```

---



We then put the disconnected network cable back in, and try pulling out the other cable, with ping results from that time frame shown in Example 11-88. Note that several seconds elapse between moving between the back of our server racks and looking at the ping output around the front, so the *icmp\_seq=xx* counter progresses by a fair number between each test.

*Example 11-88 Output of ping command after single cable switch*

---

```
64 bytes from 9.43.86.84: icmp_seq=30 ttl=64 time=0.016 ms
64 bytes from 9.43.86.84: icmp_seq=31 ttl=64 time=0.016 ms
64 bytes from 9.43.86.84: icmp_seq=32 ttl=64 time=0.011 ms
...
64 bytes from 9.43.86.84: icmp_seq=53 ttl=64 time=0.012 ms
64 bytes from 9.43.86.84: icmp_seq=54 ttl=64 time=0.015 ms
64 bytes from 9.43.86.84: icmp_seq=55 ttl=64 time=0.022 ms
```

---

Once again, it is impossible to tell where in that ping list the disconnect occurs. We now replace the network cable, so that both are connected, with ping output shown in Example 11-89.

*Example 11-89 Output of ping command before and after both cables reconnected*

---

```
64 bytes from 9.43.86.84: icmp_seq=56 ttl=64 time=0.012 ms
64 bytes from 9.43.86.84: icmp_seq=57 ttl=64 time=0.009 ms
...
64 bytes from 9.43.86.84: icmp_seq=106 ttl=64 time=0.013 ms
64 bytes from 9.43.86.84: icmp_seq=107 ttl=64 time=0.007 ms
64 bytes from 9.43.86.84: icmp_seq=108 ttl=64 time=0.007 ms
```

```
--- 9.43.86.84 ping statistics ---
108 packets transmitted, 108 received, 0% packet loss, time 107026ms
rtt min/avg/max/mdev = 0.007/0.014/0.031/0.004 ms
lochnese:~ # exit
```

---

Throughout the entire test, the looping *lssam* script output registered no change — we show a representative excerpt in Example 11-90 — all resources are in nominal states.

*Example 11-90 Output of lssam throughout entire network cable test*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
```

```

        '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
        '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
            '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
        '- Online IBM.Application:db2hadr_test-rs
            |- Online IBM.Application:db2hadr_test-rs:lead
            '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

In repeated iterations of this test, we adjusted the `1ssam` looping script to only sleep for one second, and then to not sleep at all, as there appeared to be no change to offline or Pending Offline or Pending Online status occurring. We still saw no change in resource states. Only when we used a non-bonded pair of ethernet network adapters in an equivalency definition, we saw a “Request=Lock” flag on the `db2hadr_test_rg` resource group for the duration of the `eth0` NIC being down, shown in Example 11-91. That flag disappeared when the `eth0` NIC was brought back up.

*Example 11-91 Output of `1ssam` during network cable test on unbonded NIC pair*

```

Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst0_lead_0-rs
        '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
            '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
                '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
        '- Online
IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
        '- Online
IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Request=Lock Nominal=Online
    '- Online IBM.Application:db2hadr_test-rs
        |- Online IBM.Application:db2hadr_test-rs:lead
        '- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

Overall, this is a good result from our test. We would expect there to be very little to no downtime registering at the level of `1ssam` command output. The remarkably small changes in the output of the ping command are most likely due to there being no load on the network interfaces. A heavily burdened DB2 cluster performing an OLTP role might show a different result.

### ***Testing failure of a cluster node***

This is where we experience an unplanned failure of the network on the primary node. Note that we are not actually going to stop the underlying server/node, so the default actions by the TSA HADR scripts will not perform a forced takeover to the standby. This is a safety feature to avoid the possibility of an HADR “split brain” in scenarios where one or two of the HADR communications ports cannot connect, but the external DB2 communications ports that client applications use to connect to the databases may still be working for both nodes. This is a good reason to have the tiebreaker resource share the same physical network adapters as the remote clients.

In order to have the cluster-controlled HADR performing a forced takeover on the standby, lines must be un-commented on both nodes in the `hadr_start.ksh` script provided with DB2, (or the `hadr_start.ksh` script as customized by your own site), as shown in Example 11-94.

As explained in the `hadr_start.ksh` script itself, every time a forced takeover action is carried out, an action must first be attempted to force the old primary DB2 node offline, to avoid the possibility of a “split brain”.

Leaving these lines un-commented, or re-commenting them after this test is entirely at the discretion of your site’s preferred policies, and after carrying out risk evaluation of what you would prefer an HADR cluster to do in each situation.

First we find where our `hadr_start.ksh` is located. Our test servers both show it in two locations in Example 11-92. The first location contains the DB2-supplied scripts. These are replicated to the RSCT target `sapolicies` subdirectory at the time that `regdb2salin` is executed.

#### *Example 11-92 Finding where the `hadr_start.ksh` script lives*

---

```
lochness:~ # find / -name hadr_start.ksh
/opt/ibm/db2/V9.1/ha/tsa/hadr_start.ksh
/usr/sbin/rsct/sapolicies/db2/hadr_start.ksh
```

```
lead:~ # find / -name hadr_start.ksh
/opt/ibm/db2/V9.1/ha/tsa/hadr_start.ksh
/usr/sbin/rsct/sapolicies/db2/hadr_start.ksh
```

---

To be safe, the `hadr_start.ksh` should be backed up and edited by root in the DB2 subdirectory, then copied to the RSCT target `sapolicies` subdirectory on both nodes, with commands such as those we use in Example 11-93.

*Example 11-93 Backing up the old hadr\_start.ksh and copying to RSCT target*

---

```
lead:/opt/ibm/db2/V9.1/ha/tsa # cp hadr_start.ksh hadr_start.ksh.backup
lead:/opt/ibm/db2/V9.1/ha/tsa # vi hadr_start.ksh
lead:/opt/ibm/db2/V9.1/ha/tsa # cp hadr_start.ksh
/usr/sbin/rsct/sapolicies/db2/
```

```
lochnese:/opt/ibm/db2/V9.1/ha/tsa # cp hadr_start.ksh
hadr_start.ksh.backup
lochnese:/opt/ibm/db2/V9.1/ha/tsa # vi hadr_start.ksh
lochnese:/opt/ibm/db2/V9.1/ha/tsa # cp hadr_start.ksh
/usr/sbin/rsct/sapolicies/db2/
```

---

For the TSA scripts supplied with DB2 V9.1, the appropriate section with three lines to be un-commented is as follows in Example 11-94.

*Example 11-94 Section in hadr\_start.ksh to be un-commented*

---

```
# Old primary node is still online, offline instance to prevent split-brain
# Uncomment following 3 lines to allow takeover by force
#chrg -o Offline -s "Name = '${forceRGOOfflineInCaseOfByForce?}'"
#su - ${instance_to_start?} -c "db2 takeover hadr on db ${DB2HADRDBNAME?} by force"
#logger -i -p notice -t $0 "NOTICE: Takeover by force issued, old primary instance
offlined to prevent split brain"
```

---

Leaving the section un-commented after testing is completed should be safe if you wish to have TSA perform a failover in this situation. It works admirably in our test scenario, and it is easy enough to test in your own environment before deploying into a production status.

Now we are in a position to effectively disable the primary node by removing all network connections. We could achieve the same effect by powering the server off, but this method is much quicker to restore normal functionality and test the recovery of normal cluster functionality.

We start the test with our **lssam** monitoring script. The initial system status is shown in Example 11-95 - all resources are in nominal states.

*Example 11-95 Output of lssam just before testing begins*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst0_lead_0-rs
   |- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
   |- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
   |- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
   |- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
```

```

'- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
'- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
'- Online IBM.Application:db2hadr_test-rs
| - Online IBM.Application:db2hadr_test-rs:lead
'- Offline IBM.Application:db2hadr_test-rs:lochnese

```

---

We also launch our remote DB2 connect batch file from a remote DB2 client, shown in Example 11-96. Connectivity to our HADR Primary database is working.

*Example 11-96 Launch a remote DB2 connect batch job*

---

```
C:\bat>db2 connect to testlead user db2inst0 using itsol3sj
```

Database Connection Information

```

Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TESTLEAD

```

```
C:\bat>db2 connect reset
DB20000I The SQL command completed successfully.
```

```
C:\bat>goto START
```

```
...
```

---

We now go to the primary node and physically remove both network cables. After this is done, there is no connectivity for the primary node to the outside world, and to all intents and purposes, the primary node is down. In fact, this type of node failure is the most dangerous of all scenarios, because the DB2 HADR database is in fact still up and running in the primary role on the isolated primary node.

If the HADR Standby database on the other node were to issue a forced takeover command to assume the HADR primary role, followed by the old primary node spontaneously reconnecting to the network without any cluster controls, this would cause the very “split brain” scenario we try so desperately to avoid in HADR. In such a situation, both nodes think they are HADR primary, and remote clients are unable to distinguish which one contains correct data, resulting in complete loss of data integrity in a typical OLTP environment. Even the Automatic Client Redirect feature is unable to protect remote connect requests from a “split brain”, because ACR is only designed to redirect requests to the alternate server if local connectivity is unsuccessful.

When we run `lssam` from the standby node, we see in Example 11-97 that TSA first detects the effective failure to communicate with the primary node and sets the appropriate resources into Offline and Failed offline states. Being smart enough not to assume that this means the node is actually down, TSA then issues a `Control=StartInhibited` flag against the primary node, just in case it suddenly decides to start responding again to network requests.

Most importantly, TSA accepts that the primary node is not coming back any time soon, and issues a forced takeover HADR command for the Standby node. This enables TSA to set the overall state for the Standby DB2 Instance resource group back from Offline to Online, the `db2hadr_test-rs:lochnese` HADR database resource to Online, and overall state of the HADR database resource group `db2hadr_test-rg` to Online. This means that DB2 is accepting remote database connection requests on the lochnese node, with the TEST database in HADR Primary role.

*Example 11-97 Output of lssam from Failed offline lead to HADR Primary lochnese*

---

```
lochnese:/opt/ibm/db2/V9.1/ha/tsa # lssam
Offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
|- Failed offline IBM.Application:db2_db2inst0_lead_0-rs
  '- Failed offline IBM.Application:db2_db2inst0_lead_0-rs:lead Node=Offline
  '- Failed offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Failed offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead Node=Offline
Offline IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Offline IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Offline IBM.Application:db2hadr_test-rs
    |- Failed offline IBM.Application:db2hadr_test-rs:lead Node=Offline
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
lochnese:/opt/ibm/db2/V9.1/ha/tsa # lssam
Failed offline IBM.ResourceGroup:db2_db2inst0_lead_0-rg Control=StartInhibited Nominal=Online
|- Failed offline IBM.Application:db2_db2inst0_lead_0-rs
  '- Failed offline IBM.Application:db2_db2inst0_lead_0-rs:lead Node=Offline
  '- Failed offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Failed offline IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead Node=Offline
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_lochnese_0-rs
  '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Request=Lock Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Failed offline IBM.Application:db2hadr_test-rs:lead Node=Offline
    '- Online IBM.Application:db2hadr_test-rs:lochnese
```

---

Meanwhile, we see that manually issuing new DB2 connect commands works against the primary remote database alias, shown in Example 11-98.

*Example 11-98 Output of manual DB2 connect after TSA performs HADR takeover*

---

```
C:\bat>db2 connect to testlead user db2inst1
Enter current password for db2inst1:
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST1
Local database alias = TESTLEAD
```

---

With ACR, we can even issue a successful connection against the alternate database alias shown in Example 11-99. It effectively makes no difference, both database definitions are both currently pointing to the TEST database on our new primary node, lochnese.

*Example 11-99 Output of DB2 connect to alternate database alias after HADR takeover*

---

```
C:\bat>db2 connect to testloch user db2inst1
Enter current password for db2inst1:
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST1
Local database alias = TESTLOCH
```

---

We did notice in Example 11-100 that when the primary node (lead) was initially down, the looping DB2 connect script itself had hung with no response until the TCP *keepalive* timer ran out, at which point the script was able to continue again. DB2 connections after this time were pointing to the new HADR Primary database on the other node (lochnese), because as we show in Example 11-97 on page 468, the HADR database resource on the old HADR Primary node (lead) were in a *Failed offline* state.

*Example 11-100 Output of looping DB2 connect script during node failure test*

---

```
C:\bat>db2 connect to testlead user db2inst0 using passblah
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TESTLEAD
```

```
C:\bat>db2 connect reset
DB20000I The SQL command completed successfully.
```

```
C:\bat>goto START
```

```
C:\bat>db2 connect to testlead user db2inst0 using passblah
SQL30081N A communication error has been detected. Communication protocol
being used: "TCP/IP". Communication API being used: "SOCKETS". Location
where the error was detected: "9.43.86.250". Communication function detecting
the error: "recv". Protocol specific error code(s): "10054", "*", "0".
SQLSTATE=08001
Terminate batch job (Y/N)? n
```

```
C:\bat>db2 connect reset
SQL1024N A database connection does not exist. SQLSTATE=08003
```

```
C:\bat>goto START
```

```
C:\bat>db2 connect to testlead user db2inst0 using passblah
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TESTLEAD
```

---

This Communication error / subsequent connect successful behavior is expected of all well-written DB2 applications, which should be making use of retry logic in their DB2 connections. Reducing the TCP *keepalive* value at the client to a reasonable number of seconds would also enable more timely detection of server connectivity issues, which in turn will allow client retry logic to occur more closely to when the network response failure actually occurs.

The IBM support technote in the following URL elaborately details the various options, dependencies, and recommended settings for TCP *keepalive* configuration parameters on various platforms:

<http://www-1.ibm.com/support/docview.wss?rs=203&uid=swg21231084>

We have completed the failover, it is now time to test the ability of the cluster to come back online (safely) and for both nodes to resume initial HADR roles. To achieve this, we simply plug both the network cables back in to the old primary node, and monitor what happens.

After the network cables are plugged back in again, we can successfully ping both virtual IP addresses from the remote client as shown in Example 11-101.



*Example 11-101 Output of ping command after network cable reconnect*

---

```
C:\bat>ping lead

Pinging lead [9.43.86.250] with 32 bytes of data:

Reply from 9.43.86.250: bytes=32 time=1ms TTL=61
Reply from 9.43.86.250: bytes=32 time=1ms TTL=61

Ping statistics for 9.43.86.250:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms
Control-C
^C
C:\bat>ping lochnese

Pinging lochnese [9.43.86.251] with 32 bytes of data:

Reply from 9.43.86.251: bytes=32 time=1ms TTL=61
Reply from 9.43.86.251: bytes=32 time=1ms TTL=61

Ping statistics for 9.43.86.251:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms
Control-C
```

---

At this stage, while the old primary node (lead) has come back online within the cluster, and we can log back in again to the Linux command line environment, the output from an `lssam` command in Example 11-102 shows the HADR database resource `db2hadr_test-rs:lead` is now set in an Offline state, which indicates an HADR Standby role.

*Example 11-102 Output of lssam after old primary node lead is back online*

---

```
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
```

```
'- Online IBM.Application:db2hadr_test-rs
   |- Offline IBM.Application:db2hadr_test-rs:lead
   '- Online IBM.Application:db2hadr_test-rs:lochnese
```

---

This is an extremely good result after avoiding a potential split brain scenario. Issuing a **db2pd** command from the DB2 instance user in Example 11-103 confirms that we do indeed have HADR in Connected, Peer state, with the database on the old primary node now in HADR Standby role.

*Example 11-103 Output of db2pd after old primary node lead is back online*

---

```
lead:~ # su - db2inst0
db2inst0@lead:~> db2pd -db test -hadr

Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:15:23

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Standby Peer              Sync      0                  0

ConnectStatus ConnectTime              Timeout
Connected    Mon Apr 21 21:31:01 2008 (1208838661) 15

LocalHost           LocalService
lead                55101

RemoteHost          RemoteService  RemoteInstance
lochnese            55102         db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000007.LOG 0      0x0000000002BA0000

StandByFile StandByPg StandByLSN
S0000007.LOG 0      0x0000000002BA0000
```

---

If any HADR catchup pending state or log gap existed (not using SYNCMODE or database not in Peer state at time of node failure), then manual intervention would be required here to re-establish the HADR pairing, as the old HADR Primary database would not be able to successfully start as an HADR Standby. It would contain log records which the current HADR Primary database did not know about, and would not be able to re-synchronize. This situation occurs as a result of data loss occurring between Primary and Standby nodes at the time of Primary node failure, which can be a natural symptom of using the following HADR synchronization modes:

HADR ASYNC mode can result in data loss even if the pair are in PEER state when the log records from the Primary carried over the network do not all make it to the standby at the time of a failure on the primary node.

HADR NEARSYNC mode should not result in data loss if the pair are in Peer state and the Standby does not fail while processing outstanding log records.

In any HADR failover scenario where the database pair are not in Peer state, loss of data will occur, so it is up to your individual site as to whether those data losses are preferable to having an extended outage to bring the old primary node back online.

The following link in the DB2 V9.5 Information Center provides valuable information on the effects of an HADR failover/forced takeover and required considerations in various scenarios, specifically those which may incur data loss:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0011835.html>

Allowing an HADR takeover by force in non-peer state is enabled by un-commenting the specified lines from this paragraph in the `hadr_start.ksh` script (Example 11-104).

*Example 11-104 Output of db2pd after old primary node lead is back online*

---

```
# Uncomment following 3 lines to allow takeover even in case of non Peer Standby w/ old
Primary machine Online
#chrg -o Offline -s "Name = '${forceRGOOfflineInCaseOfByForce?}'"
#su - ${instance_to_start?} -c "db2 takeover hadr on db ${DB2HADRDBNAME?} by force "
#logger -i -p notice -t $0 "NOTICE: Takeover by force issued, old primary instance
offlined to prevent split brain"
```

---

Back to our testing scenario, all we should need to do now to get our cluster back to the original state is to issue a resource group request. This is done with a **rgreq** command as shown in Example 11-105. The output from **lssam** shows the HADR resource group being set to Pending Online while the request is being performed, and finally, the `db2hadr_test-rs:lead` resource is set to HADR Primary (Online), and the `db2hadr_test-rs:lochnese` is set to Standby (Offline).

*Example 11-105 Output of lssam after issuing rgreq command to switch HADR roles*

---

```
lead:~ # rgreq -o move -n lochnese db2hadr_test-rg
Action on resource group "db2hadr_test-rg" returned Token
"0x8d04fb624c0804b1516e0d48596f0500" .
lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Pending online IBM.ResourceGroup:db2hadr_test-rg Request=Move Nominal=Online
  '- Pending online IBM.Application:db2hadr_test-rs
    |- Pending online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese

lead:~ # lssam
Online IBM.ResourceGroup:db2_db2inst0_lead_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst0_lead_0-rs
    '- Online IBM.Application:db2_db2inst0_lead_0-rs:lead
  '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst0_lead_0-rs_ip:lead
Online IBM.ResourceGroup:db2_db2inst1_lochnese_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_lochnese_0-rs
    '- Online IBM.Application:db2_db2inst1_lochnese_0-rs:lochnese
  '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip
    '- Online IBM.ServiceIP:db2_db2inst1_lochnese_0-rs_ip:lochnese
Online IBM.ResourceGroup:db2hadr_test-rg Nominal=Online
  '- Online IBM.Application:db2hadr_test-rs
    |- Online IBM.Application:db2hadr_test-rs:lead
    '- Offline IBM.Application:db2hadr_test-rs:lochnese
```

---

One final db2pd command in Example 11-106 confirms that we are back to normal, with the TEST database on node lead in HADR Primary role, in Connected, Peer state.

*Example 11-106 Output of db2pd after rgreq command to switch HADR Primary role*

---

```
db2inst0@lead:~> db2pd -d test -hadr
```

```
Database Partition 0 -- Database TEST -- Active -- Up 0 days 00:20:22
```

HADR Information:

| Role    | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Primary | Peer  | Sync     | 0                | 0                    |

| ConnectStatus | ConnectTime                           | Timeout |
|---------------|---------------------------------------|---------|
| Connected     | Mon Apr 21 21:31:01 2008 (1208838661) | 15      |

| LocalHost | LocalService |
|-----------|--------------|
| lead      | 55101        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| lochnese   | 55102         | db2inst1       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000007.LOG | 0         | 0x0000000002BA0000 |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000007.LOG | 0         | 0x0000000002BA0000 |

---

This concludes our testing of DB2 V9.1 HADR in a TSA-controlled cluster.





# Configuring clusters using the DB2 9.5 High Availability Feature

In this chapter we discuss the DB2 high availability (HA) feature, which integrates High Availability clustering functionality as an integral part of the DB2 solution. This is a valuable level of abstraction from underlying clustering software mechanisms. Critical to the success of this feature is an automated way to create and configure the HA cluster. Example scenarios are provided, discussing the complete HA lifecycle that includes describing the architecture, describing the cluster configuration, then performing basic testing and administration. We present a broad cross section of the automation scenarios possible, including single partition shared disk, single partition data replication automation (HADR), and multi-partition failover automation (DPF).

References for this chapter are available in the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>

Also see the IBM White Paper, *Automated Instance Failover using the IBM DB2 High Availability Instance Configuration Utility (db2haicu)*, at:

[http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/db2/papers/db2\\_db2haicu.pdf](http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/db2/papers/db2_db2haicu.pdf)

## 12.1 db2haicu

DB2 High Availability Instance Configuration Utility (db2haicu) is a text based utility that can be used to configure and administer your highly available databases in a clustered environment. It collects information about the database instance, cluster environment, and cluster manager by querying the system. Information can be supplied through parameters to the db2haicu call, an input file, or runtime at db2haicu prompts.

### 12.1.1 Prerequisites

Before running db2haicu, first there is a set of tasks to be performed by a user with root authority and database manager instance owner.

A user with root authority must initialize db2haicu on all machines that are to be added to the cluster domain by running the following command:

```
preprnode
```

This prepares security on the node on which the command is run so it can be defined in a peer domain. It allows for peer domain operations to be performed on the node and must be run before the node can join a peer domain.

For example:

```
/usr/sbin/preprnode <nodename>
```

This command only needs to be run once per node and not for every DB2 instance that is made highly available.

A database manager instance owner must perform the following tasks:

- ▶ Synchronize /etc/services files on all machines that are to be added to the cluster.
- ▶ Run the db2profile script for the database manager instance to be used to create the cluster domain.
- ▶ Start the database manager using the **db2start** command.

There are some tasks specific to DB2 High Availability Disaster Recovery (HADR). If HADR is already configured on your system, perform the following tasks:

- ▶ Ensure that all HADR databases are started in their respective primary and standby database roles, and that all HADR primary-standby database pairs are in the peer state.



- ▶ Configure the HADR\_PEER\_WINDOW database configuration parameter for all HADR databases to a value of at least 120 seconds.
- ▶ Disable the DB2 fault monitor.

For a partitioned database environment, there is a specific task to be done prior to configuring it for high availability:

- ▶ Configure the DB2\_NUM\_FAILOVER\_NODES registry variable on all machines that are to be added to the cluster domain. This specifies the number of additional database partitions that might need to be started on a machine in the event of failover.

When all these tasks are done, a database manager instance owner can use db2haicu to perform cluster configuration and administration operations.

## 12.1.2 Usage

The db2haicu utility takes in user input regarding the software and hardware environment of a DB2 instance, and configures the instance for high availability using the SA MP Cluster Manager. During this configuration process, all necessary resources, dependencies, and equivalencies are automatically defined to SA MP.

### Syntax

The syntax for db2haicu is as follows:

```
db2haicu [ -f <XML-input-file-name> ]
          [ -disable ]
          [ -delete [ dbpartitionnum <db-partition-list> |
                    hadrdb <database-name> ] ]
```

The parameters that you pass to the **db2haicu** command are case-sensitive, and must be in lowercase.

**-f <XML-input-file-name>**

You can use the **-f** parameter to specify your cluster domain details in an XML input file.

For example, if you have an XML file called DPF.xml that has all cluster definitions, you can run the following command to create a cluster:

```
db2haicu -f DPF.xml
```

**-disable**

To disable the HA configuration for a particular instance, this option can be used. After issuing this command, the system does not respond to any failures and all resource groups for the instance are locked.

## -delete

You can use the `-delete` parameter to delete resource groups for the current database manager instance. If you do not use either the `dbpartitionnum` parameter or the `hadrdb` parameter, then `db2haicu` removes all the resource groups associated with the current database manager instance.

`dbpartitionnum <db-partition-list>`

You can use the `dbpartitionnum` parameter to delete resource groups that are associated with the database partitions listed in `<db-partition-list>`. It is a comma-separated list of numbers identifying the database partitions.

`hadrdb <database-name>`

You can use the `hadrdb` parameter to delete resource groups that are associated with a specific HADR database, `<database-name>`. If there are no resource groups left in the cluster domain after `db2haicu` removes the resource groups, then `db2haicu` also removes the cluster domain.

After a cluster domain is removed, you can reconfigure a database manager instance for HA by running `db2haicu` again either in interactive mode or batch mode.

For example, to remove all the resource groups associated with the current database manager instance, you can run the following command:

```
db2haicu -delete
```

## Modes

To configure a database manager instance for high availability, you can run `db2haicu` in one of the following modes:

► Interactive mode:

When you invoke `db2haicu` without specifying an XML input file with the `-f` parameter, it runs in interactive mode, displays information, and prompts for information in a text-based format.

To run `db2haicu` in interactive mode, call the **db2haicu** command without the `-f <input-file-name>` parameter.

► Batch mode with an XML input file:

When you specify an XML input file with the **db2haicu** command, it runs in batch mode and uses the configuration details provided in the file. This option is very useful when you must perform configuration for multiple database partitions for high availability.

To configure your clustered environment for the current database manager instance using db2haicu and an input file that you create called db2haicu-sample.xml, use the following command:

```
db2haicu -f db2haicu-sample.xml
```

There is a set of sample XML input files located in the samples subdirectory of the sqllib directory that you can modify and use with db2haicu to configure your clustered environment.

► **Startup mode:**

When db2haicu is run for the first time for a given database manager instance, db2haicu operates in startup mode. It examines your database manager instance and your system configuration, and searches for an existing cluster domain. If there is no cluster domain created and configured for the instance, db2haicu immediately begins the process of creating and configuring a new cluster domain by prompting for information such as a name for the new cluster domain and the host name of the current machine.

If you create a cluster domain, but do not complete the task of configuring the cluster domain, then the next time you run db2haicu, it resumes the task of configuring the cluster domain. After a cluster domain is configured, db2haicu runs in maintenance mode.

A cluster domain is a model that contains information about your database and cluster elements such as databases, mount points, and failover policies. db2haicu uses the information in the cluster domain to manage configuration and maintenance of database and cluster elements.

► **Maintenance mode:**

When db2haicu is run and there is already a cluster domain created for the current database manager instance, db2haicu operates in maintenance mode. In this mode, it presents you with a list of configuration and administration tasks that you can perform, including these:

- Add or remove cluster nodes.
- Add or remove a network interface.
- Add or remove a highly available database.
- Add or remove a mount point.
- Add or remove an IP address.
- Add or remove a non-critical path.
- Move DB2 database partitions and HADR databases for scheduled maintenance.
- Change failover policy for this instance.
- Create a new quorum device for the domain.
- Destroy the domain.
- Exit.

### 12.1.3 Recommendations

Consider the following list of recommendations for configuring your cluster and your database manager instances when using db2haicu:

- ▶ Network time protocol (NTP): In the case of HADR and DPF cluster configurations, the time and dates on all nodes should be synchronized as closely as possible. This is absolutely critical to ensure smooth failover.

For an automatic instance failover, we recommend (but it is not mandatory) that the time and dates on cluster nodes be synchronized.

Refer to your operating system documentation for information on how to configure NTP for your system.

- ▶ For an automatic instance failover cluster configuration, when you add new mount points for the cluster by adding entries to /etc/fstab on all cluster nodes, use the noauto option to prevent the mount points from being automatically mounted on more than one machine in the cluster. For example:

```
# LUN          Mount Point      FileSystem Type  Automount
/dev/sdd       /shared_home     ext3             noauto
```

- ▶ Any maintenance work can be performed by disabling the HA configuration using **db2haicu -disable** without worrying about cluster manager intervention. After issuing this command, the system does not respond to any failures, and all resource groups for the instance are locked.

### 12.1.4 Troubleshooting

You can investigate and diagnose db2haicu errors using the database manager diagnostic log, db2diag.log, and the db2pd tool. There is no separate diagnostic log that db2haicu uses to log all errors and warnings.

If db2haicu fails with an error while you are creating and configuring a new cluster domain, you must perform the following steps:

- ▶ Remove the resource groups of the partially created cluster domain by running **db2haicu** using the **-delete** parameter
- ▶ Recreate the new cluster domain by calling the **db2haicu** utility again.

## 12.2 DB2 HADR configuration for automatic failover with on AIX

This section describes how to configure automatic DB2 HADR takeover with Tivoli System Automation (TSA) using the **db2haicu** command in an AIX environment. The manual configuration procedure is described in 11.3, “Automating HADR takeover with TSA on AIX” on page 377.

In an HADR with TSA environment, TSA facilitates the detection of failures and automatically executes HADR takeover from one system to another in the cluster after a hardware or software failure. In a TSA cluster, HADR is placed under TSA control. For more details regarding TSA, refer to Chapter 8, “DB2 with TSA” on page 233.

### 12.2.1 Architecture

Before you set up an HADR with TSA environment, you must plan the cluster environment. Consider the following items:

- ▶ Physical nodes:
  - Decide on the primary and the secondary nodes. The primary node (or service node) regularly provides service to clients. The HADR primary database runs on this node. The role of both nodes can be changed in response to system failover or planned takeover issued by administrators.
  - Confirm software requirements on each node.
- ▶ Network:
  - Network interface that provides client access on each node. This is one of the TSA controlled resources.
  - Have one network card for a TSA tiebreaker. We recommend that you have a separate network for HADR to avoid the interference of HADR log transfer.
- ▶ TSA configuration:

Define a cluster domain name that includes the resources of virtual IP (service IP), DB2 instance, and HADR database. The **db2haicu** commands automatically create the cluster domain and resource groups.
- ▶ HADR configuration:

Define the HADR configuration. The **db2haicu** command does not automatically create the HADR configuration. Refer to Chapter 5, “Automatic client reroute” on page 121 to define HADR configuration.

► Scripts:

DB2 9.5 automatically installs the TSA scripts used for start, stop, and monitoring DB2 resources.

## Lab environment

Figure 12-1 shows the configuration of HADR with TSA in our lab environment.

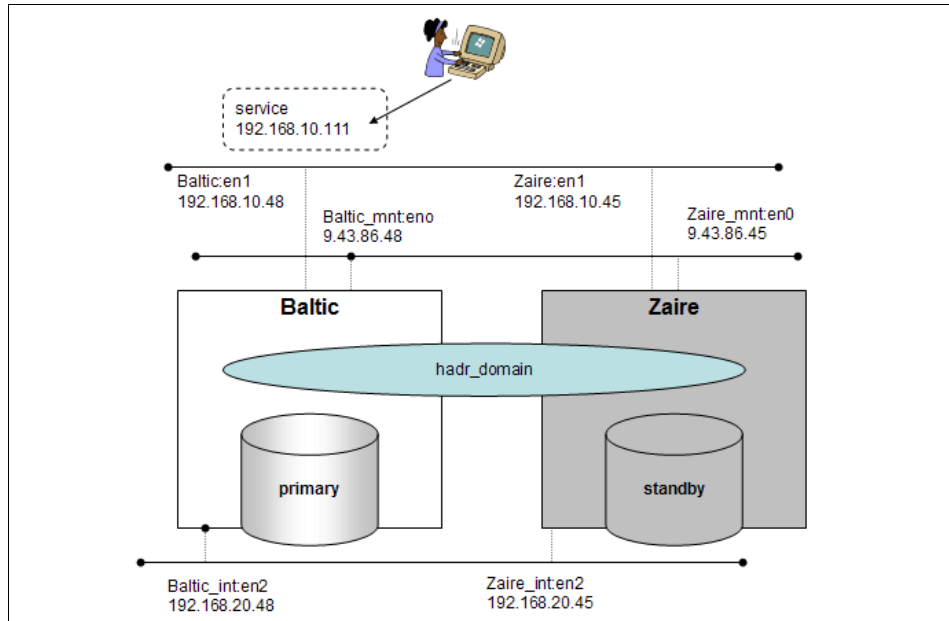


Figure 12-1 HADR with TSA lab environment

The planning for our lab environment is as follows:

► Physical node configurations:

- Two physical nodes named **Baltic** and **Zaire** are defined in the cluster.
- **Baltic** is designated as *service node* and has the HADR primary database.
- **Zaire** is the *standby* node on which the HADR standby database resides.
- The software configuration used to set up the lab environment includes:
  - AIX 5.3 Technology Level 7
  - DB2 9.5 Enterprise Fix Pack 1
  - Tivoli System Automation for Multiplatforms (SA MP) Base Component V2.2

- ▶ Network configurations:
  - One ethernet network interface (en1) on each node is provided for client access, under the control of TSA. The service address for the clients is added on one network interface as follows:
    - Primary node (Baltic)
    - en1: 192.168.10.48 (255.255.252.0)
    - Standby node (Zaire)
    - en1: 192.168.10.45 (255.255.252.0)
  - One ethernet network interface (en2) is dedicated to HADR communications:
    - Primary node (Baltic)
    - en2: 192.168.20.48 (255.255.252.0)
    - Standby node (Zaire)
    - en2: 192.168.20.45 (255.255.252.0)
  - One ethernet network is configured for a TSA tiebreaker. Refer to 8.1.2, “Terminology of TSA” on page 236 for the detail description of the TSA tiebreaker.

**Note:** For HADR in a TSA cluster, shared disks are not necessary because the primary and standby databases are independent databases, which can reside in separate storage devices.

- ▶ TSA configuration:
  - Cluster Domain named *hadr\_domain* is configured, which includes the resources of virtual IP (service IP), DB2 instance, and HADR database.
  - In our lab, we configured a virtual IP in the TSA cluster domain.
- ▶ HADR configuration:
  - Each node has the DB2 instance named *db2inst1*. Instance names do not have to be identical on both the nodes.
  - Each instance has the database named *SAMPLE*.
  - Configured dedicated network for HADR communication.

## 12.2.2 Configuration

In this section, we provide the steps to configure an automated HADR takeover environment with TSA.

## HADR setup

To configure the HADR database pair, complete the following steps:

1. For new systems, create a DB2 instance on both nodes. We created *db2inst1*.
2. Check that proper entries are configured in */etc/hosts* and */etc/services* files.
3. For new systems, create a database on the primary node. We created *SAMPLE*.
4. Back up the primary database and restore the image on the standby node.
5. Configure HADR parameters properly on both the databases.
6. Start HADR on the standby database, and then start HADR on the primary database.
7. Check that both databases can communicate with each other in the peer state using *db2pd*.

See Chapter 3, “HADR setup” on page 47 for the step-by-step configuration of HADR.

Figure 12-2 shows the entries of the services and hosts of the HADR configuration in our lab environment.

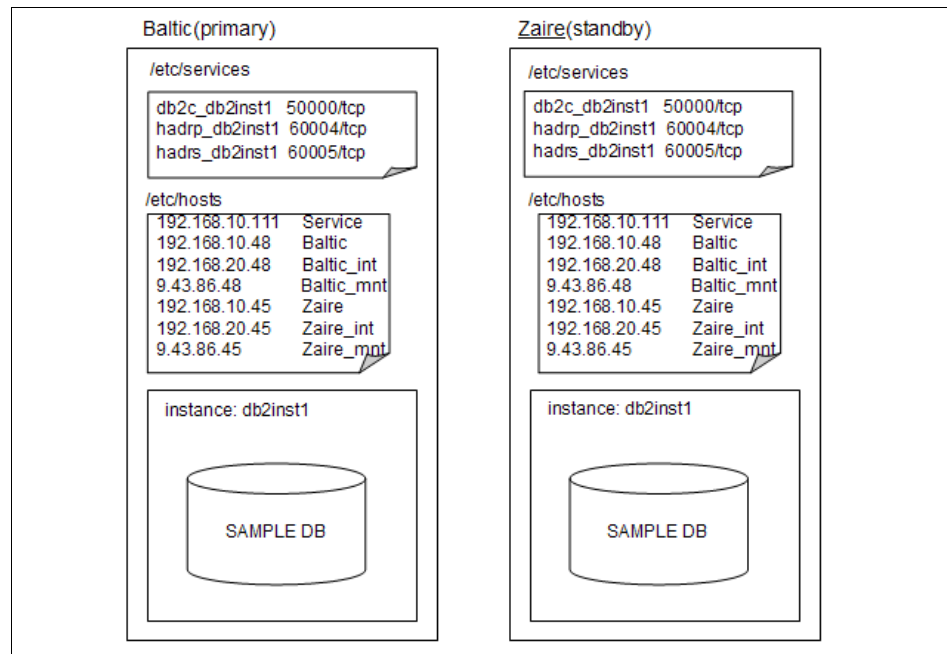


Figure 12-2 Configuration of HADR



## Checking the TCP/IP network connection

Before you configure TSA, check that the network is configured properly. To check network configurations for the cluster, complete the following steps:

1. Check whether the IP addresses are configured on network interfaces using the command:

```
netstat -in | grep -v link
```

Example 12-1 shows the configuration of the IP addresses on Baltic.

### *Example 12-1 IP address configuration*

---

```
(P)(S) #netstat -in | grep -v link
Name Mtu Network Address
en0 1500 9.43.84 9.43.86.48
en1 1500 192.168.8 192.168.10.48
en2 1500 192.168.20 192.168.20.48
lo0 16896 127 127.0.0.1
```

---

2. Check whether the `/etc/hosts` file has all the entries of IP addresses:

Example 12-2 shows the content of our hosts files.

### *Example 12-2 hosts file content*

#### **(P)(S)#vi /etc/hosts**

---

```
192.168.10.111 Service ## Virtual IP address

192.168.10.48 Baltic ## TSA N/W interface for Virtual IP on Baltic
192.168.20.48 Baltic_int ## HADR N/W interface on Baltic
9.43.86.48 Baltic_mnt ## For Maintenance N/W interface on Baltic

192.168.10.45 Zaire ## TSA N/W interface for Virtual IP on Zaire
192.168.20.45 Zaire_int ## HADR N/W interface on Zaire
9.43.86.45 Zaire_mnt ## For Maintenance N/W interface on Zaire
```

---

3. Verify that name resolution is working well by using the `host` command. If something is wrong, check and modify the `/etc/hosts` file:

```
(P)(S)#host Baltic
Baltic is 192.168.10.48
```

## Setting up the cluster domain of TSA by db2haicu

The main steps for this setup are as follows:

1. Initial configuration
2. Configure cluster domain and resource by db2haicu

Next we provide the detailed steps for this setup.

### **Initial configuration**

Run the **preprnode** command as root to prepare the local node for joining the domain:

```
(P)(S) # preprnode Baltic Zaire
```

### **Configure cluster domain and resource by db2haicu**

In this section, we demonstrate the cluster configuration using the *db2haicu XML file setup mode*. For configuration with db2haicu interactive setup mode, refer to 12.3, “DB2 HADR configuration for automated failover on Linux” on page 508.

Make sure that all the nodes in to be configured into the TSA cluster domain can see each other in the network and can communicate with each other.

Issue the **db2haicu** command to create the cluster domain and resource group on either node:

1. Create the XML file:

The db2haicu XML file contains all the information that db2haicu needs to know in order to make a DB2 HADR instance cooperate with TSA. DB2 provides sample XML files located in the `sqlib/samples/ha/xml` directory. These sample XML files can also be found at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0052514.html>

Example 12-3 illustrates a sample db2haicu XML file.

#### **Example 12-3 The example of XML file on standby node**

---

```
<?xml version="1.0" encoding="UTF-8"?>
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="db2ha.xsd" clusterManagerName="TSA" version="1.0">
  <ClusterDomain domainName="hadr_dom">
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="192.168.10.51"/>
    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
      <Interface interfaceName="en1" clusterNodeName="Baltic">
        <IPAddress baseAddress="192.168.10.48" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
      </Interface>
      <Interface interfaceName="en1" clusterNodeName="Zaire">
        <IPAddress baseAddress="192.168.10.45" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
      </Interface>
    </PhysicalNetwork>
    <PhysicalNetwork physicalNetworkName="db2_private_network_0"
physicalNetworkProtocol="ip">
```

```

        <Interface interfaceName="en2" clusterNodeName="Baltic">
            <IPAddress baseAddress="192.168.20.48" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
        </Interface>
        <Interface interfaceName="en2" clusterNodeName="Zaire">
            <IPAddress baseAddress="192.168.20.45" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
        </Interface>
    </PhysicalNetwork>
    <ClusterNode clusterNodeName="Baltic"/>
    <ClusterNode clusterNodeName="Zaire"/>
</ClusterDomain>
<FailoverPolicy>
    <HADRFailover></HADRFailover>
</FailoverPolicy>
<DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
        </DB2Partition>
    </DB2PartitionSet>
<HADRDBSet>
    <HADRDB databaseName="SAMPLE" localInstance="db2inst1" remoteInstance="db2inst1"
localHost="Zaire" remoteHost="Baltic" />
    <VirtualIPAddress baseAddress="192.168.10.111" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
</HADRDBSet>
</DB2Cluster>

```

---

The existing values in this XML file can be replaced to reflect your own configuration and environment. The XML elements in the configuration files are as follows:

- The *<ClusterDomain>* element covers all cluster-wide information. This includes the nodes in the cluster domain; the network equivalencies (groups of networks that can fail over for one another); and the quorum device (tie-breaking mechanism).
  - The *<Quorum>* sub-element of the *ClusterDomain* element specifies the quorum device for the cluster domain.
  - The *<PhysicalNetwork>* sub-element of the *ClusterDomain* element includes all network information. This includes the name of the network and the network interface cards contained in it. We define our single public network and private network using this element.
  - The *<ClusterNode>* sub-element contains information about a particular node in the cluster.
- The *<FailoverPolicy>* element specifies the failover policy that the cluster manager should use with the cluster domain. Select one of the following failover policies for the cluster manager to follow if there is a failure in the cluster domain (we define *HADRFailover* in our case):

- RoundRobin: When you are using a round robin failover policy, if there is a failure associated with one computer in the cluster domain, the database manager restarts the work from the failed cluster domain node on any other node in the cluster domain.
  - Mutual: To configure a mutual failover policy, you associate a pair of computers in the cluster domain as a system pair. If there is a failure on one of the nodes in the pair, the database partitions on the failed node fail over to the other node in the pair. Mutual failover is only available when you have multiple database partitions (DPF).
  - NPlusM: When you are using an N Plus M failover policy, if there is a failure associated with one computer in the cluster domain, the database partitions on the failed node fail over to any other node that is in the cluster domain. N Plus M failover is only available when you have multiple database partitions (DPF).
  - LocalRestart: When you use a local restart failover policy, if there is a failure on one of the computers in the cluster domain, the database manager restarts the database in place (or locally) on the same node that failed.
  - HADRFailover: When you configure an HADR failover policy, you are enabling HADR feature to manage failover. If an HADR primary database fails, the database manager moves the workload from the failed database to the HADR standby database.
  - Custom: When you configure a custom failover policy, you create a list of nodes in the cluster domain onto which the database manager can fail the resources over. If a node in the cluster domain fails, the database manager moves the workload from the failed node to one of the nodes in the list that you specified
- The `<DB2PartitionSet>` element covers the DB2 instance information. This includes the current DB2 instance name and the DB2 partition number.
  - The `<HADRDBSet>` element covers the HADR database information. This includes the primary node name, standby node name, primary instance name, standby instance name, and the virtual IP address associated with the database.

For a more detailed description of the XML files, refer to:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0053130.html>

## 2. Execute the `db2haicu` command.

`db2haicu` must be run first on the standby instance and then on the primary instance for the configuration to complete:

```
db2haicu -f XMLfilepath
```

Example 12-4 shows the sample output of `db2haicu` with XML on the standby node.

*Example 12-4 Sample output of db2haicu with XML on the standby*

---

**(S) \$ db2haicu -f db2hadr.xml**

Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called `db2diag.log`. Also, you can use the utility called `db2pd` to query the status of the cluster domains you create.

For more information about configuring your clustered environment using `db2haicu`, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

`db2haicu` determined the current DB2 database manager instance is `db2inst1`. The cluster configuration that follows will apply to this instance.

`db2haicu` is collecting information on your current setup. This step may take some time as `db2haicu` will need to activate all databases for the instance to discover all paths ...

Creating domain `hadr_dom` in the cluster ...

Creating domain `hadr_dom` in the cluster was successful.

Configuring quorum device for domain `hadr_dom` ...

Configuring quorum device for domain `hadr_dom` was successful.

Adding network interface card `en1` on cluster node `Baltic` to the network `db2_public_network_0` ...

Adding network interface card `en1` on cluster node `Baltic` to the network `db2_public_network_0` was successful.

Adding network interface card `en1` on cluster node `Zaire` to the network `db2_public_network_0` ...

Adding network interface card `en1` on cluster node `Zaire` to the network `db2_public_network_0` was successful.

Adding network interface card `en2` on cluster node `Baltic` to the network `db2_private_network_0` ...

Adding network interface card `en2` on cluster node `Baltic` to the network `db2_private_network_0` was successful.

Adding network interface card `en2` on cluster node `Zaire` to the network `db2_private_network_0` ...

Adding network interface card `en2` on cluster node `Zaire` to the network `db2_private_network_0` was successful.

Adding DB2 database partition 0 to the cluster ...

Adding DB2 database partition 0 to the cluster was successful.

The HADR database `SAMPLE` has been determined to be valid for high availability.

However, the database cannot be added to the cluster from this node because `db2haicu` detected this node is the standby for the HADR database `SAMPLE`. Run `db2haicu` on the primary for the HADR database `SAMPLE` to configure the database for automated failover.

All cluster configurations have been completed successfully. `db2haicu` exiting ...

---

Example 12-5 shows the sample output of **db2haicu** with XML on the primary node.

*Example 12-5 Sample output of db2haicu with XML on the primary*

---

**(P) \$ db2haicu -f db2hadr.xml**

Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst1. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

Configuring quorum device for domain hadr\_dom ...

Configuring quorum device for domain hadr\_dom was successful.

**The network adapter en1 on node Baltic is already defined in network db2\_public\_network\_0 and cannot be added to another network until it is removed from its current network.**

**The network adapter en1 on node Zaire is already defined in network db2\_public\_network\_0 and cannot be added to another network until it is removed from its current network.**

**The network adapter en2 on node Baltic is already defined in network db2\_private\_network\_0 and cannot be added to another network until it is removed from its current network.**

**The network adapter en2 on node Zaire is already defined in network db2\_private\_network\_0 and cannot be added to another network until it is removed from its current network.**

Adding DB2 database partition 0 to the cluster ...

Adding DB2 database partition 0 to the cluster was successful.

Adding HADR database SAMPLE to the domain ...

Adding HADR database SAMPLE to the domain was successful.

All cluster configurations have been completed successfully. db2haicu exiting ...

---

**Note:** The messages regarding the networks (bold text) encountered on the primary node can be safely ignored. These messages appear because we have already defined the public network to db2haicu through the standby node.

The HADR configuration is completed right after db2haicu runs the XML file on the primary instance. Use the **lssam**, **lsrpdomain**, and **lsrpnode** commands to see the resources created during this process.

Example 12-6 lists the domain cluster resource status with **1ssam**.

*Example 12-6 List domain cluster resource status with 1ssam*

---

**(P) #1ssam**

```
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
    '- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

In Example 12-7, cluster domain status is listed with **1srpdomain**.

*Example 12-7 List cluster domain status*

---

**(P) # 1srpdomain**

| Name        | OpState | RSCTActiveVersion | MixedVersions | TSPort | GSPort |
|-------------|---------|-------------------|---------------|--------|--------|
| hadr_domain | Online  | 2.4.8.3           | No            | 12347  | 12348  |

---

Example 12-8 lists the nodes in the cluster with **1srpnode**.

*Example 12-8 List of nodes in the cluster*

---

**(P) # 1srpnode**

| Name   | OpState | RSCTVersion |
|--------|---------|-------------|
| baltic | Online  | 2.4.8.3     |
| zaire  | Online  | 2.4.8.3     |

---

## Resource groups topology

After the **db2haicu** tool is run successfully on both standby and primary instances, the setup is complete.

The relationships among the resources are illustrated in Figure 12-3. The following list shows how the resources illustrated in this figure correspond to the resources listed in the **1ssam** command shown in Example 12-6:

- ▶ Primary DB2 instance resource group: *db2\_db2inst1\_Baltic\_0-rg* (**1ssam**)
  - Member resources:
    - Primary DB2 resource: *db2\_db2inst1\_Baltic\_0-rs* (**1ssam**)

- ▶ Standby DB2 instance resource group: *db2\_db2inst1\_Zaire\_0-rg (1 ssam)*
  - Member resources:
    - Standby DB2 resource: *db2\_db2inst1\_Zaire\_0-rs*
- ▶ HADR database resource group: *db2\_db2inst1\_db2inst1\_HADRDB-rg (1 ssam)*
  - Member resources:
    - HADR DB resource: *db2\_db2inst1\_db2inst1\_HADRDB-rs (1 ssam)*
    - Virtual IP address resource: *db2ip\_192\_168\_10\_111-rs (1 ssam)*

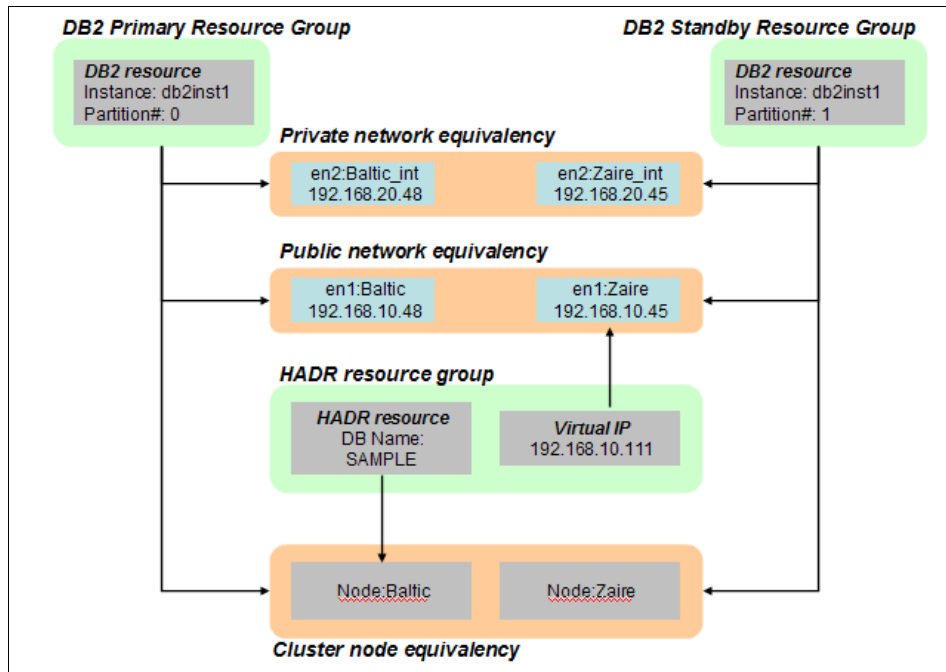


Figure 12-3 Resource groups created for a multiple network HADR topology

## Deleting the domain

The `db2haicu` option `-delete` removes an entire HA configuration from a system and deletes all resources in the cluster. If the other instances are using the domain at the time, the domain is deleted as well.

As a good practice, we recommend running `db2haicu` with the `delete` option on an instance before it is made highly available. This makes sure that the setup is started from scratch and there are no residuals from the previous build. For example, when running `db2haicu` with an XML file, any invalid attribute in the file causes `db2haicu` to exit with a non-zero error code. Before `db2haicu` is run again



with the corrected XML file, you can run the **delete** option to make sure that any temporary resources created during the initial run are cleaned up.

Note that the `db2haicu -delete` option leaves the DB2 instances and the HADR replication unaffected. That is, it does not stop the DB2 instances of HADR replications. However, any IP addresses that were highly available are removed and are no longer presented after the **db2haicu -delete** command completes.

Example 12-9 shows a sample output of the **db2haicu -delete** command.

*Example 12-9 Output of db2haicu -delete*

---

**(P) (S) \$db2haicu -delete**

Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called `db2diag.log`. Also, you can use the utility called `db2pd` to query the status of the cluster domains you create.

For more information about configuring your clustered environment using `db2haicu`, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

`db2haicu` determined the current DB2 database manager instance is `db2inst1`. The cluster configuration that follows will apply to this instance.

When you use `db2haicu` to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with `db2haicu`' in the DB2 Information Center. `db2haicu` is searching the current node for an existing active cluster domain ...

`db2haicu` found a cluster domain called `Baltic` on this node. The cluster configuration that follows will apply to this domain.

Deleting the domain `Baltic` from the cluster ...

Deleting the domain `Baltic` from the cluster was successful.

All cluster configurations have been completed successfully. `db2haicu` exiting ...

---

### 12.2.3 Administration

From time to time, you need to plan system outage for some maintenance activities such as software upgrade or recycle DB2 instance for changing non-dynamic database manager parameters. In this section, we demonstrate how to perform some DB2, OS, and TSA operations manually for the planned outage or maintenance.

When you complete the DB2 HADR with TSA setup, you can perform these operations as some simple tests to verify your configuration.

## Manual DB2 instance operations

Next we discuss how to stop and start DB2 on the primary and the standby nodes for the planned outage or maintenance.

### ***Issue the db2stop and db2start commands on the standby***

You can issue the **db2stop** and **db2start** commands on the standby node without impacting the activities taking place at the primary database.

The **db2stop force** command stops the instance and causes HADR replication to halt:

```
(S) $db2stop force
```

After that, the resource group of the standby instance (db2\_db2inst1\_Zaire\_0-rg) is in *Pending Online* state. And we can see the *LOCK* status is placed on the HADR resource group and the standby instance resource group. This lock indicates that the HADR databases are no longer in the peer state.

Example 12-10 illustrates the effect of the **db2stop force** command issued on the standby instance.

#### *Example 12-10 Issam output after stopping DB2 on standby*

---

```
(P)(S) #lssam
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Pending online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Request=Lock
Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Offline IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock
Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
    '- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

To recover from this state, you start the standby DB2 instance and activate the database again use the following commands:

```
(S) $db2start;db2 activate db sample
```

### ***Issue the db2stop and db2start commands on the primary***

Issue the **db2stop force** command on the primary instance causes the connection from client broken and HADR replication to halt.

```
(P) $db2stop force
```

Example 12-11 illustrates the effect of the **db2stop force** command issued on the primary instance. The takeover does not occur because that is a planned operation.

*Example 12-11 Issam output after stopping DB2 on primary*

---

```
(P) (S) #lssam
Offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Request=Lock Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Offline IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock
Nominal=Online
  |- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
      '- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
      '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

To recover from this state, you start the primary DB2 instance and activate the database again.

```
(P) $db2start;db2 activate db sample
```

**Issue the takeover hadr command on the standby**

There might be situations when a DBA wants to perform a manual takeover to switch database roles. The safest way to accomplish this is to issue the **HADR takeover hadr** command from the command line without the **by force** option.

Log on to the standby node and run the **takeover hadr on db dbname** command to perform a manual takeover. For example:

```
(S) $db2 takeover hadr on db sample
```

After the takeover has been completed successfully, the **lssam** commands reflect the changes. The virtual IP (service IP) address is also moved to the new primary node as part of the takeover process. See Example 12-12.

*Example 12-12 Issam output after takeover*

---

```
(P) (S) #lssam
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
```

```

Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
   |- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
   '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
'- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
   |- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
   '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

---

## Manual OS operations

We discuss OS and TSA operations in this section for the planned outage or maintenance.

### ***Stop the standby node***

If you need to stop TSA on the standby node for maintenance, follow these steps:

1. Issue the **db2stop force** command on the standby instance:  
(S) \$db2stop force
2. Issue the **stoprnode hostname** command as root user on the primary node. If you execute stoprnode command on the standby, you cannot issue any TSA or RSCT operations on the standby node. So you issue these commands on the primary node:  
(P) #stoprnode Zaire  
(P) #lsrpnode  
Name OpState RSCTVersion  
Zaire Offline 2.4.8.3  
Baltic Online 2.4.8.3
3. Issue the **lssam** command to observe the state of the resources as shown in Example 12-13.

#### *Example 12-13 lssam output after stoprnode*

---

```

(P) #lssam
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
   '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
      '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Failed offline IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Request=Lock Nominal=Online
   '- Failed offline IBM.Application:db2_db2inst1_Zaire_0-rs
      '- Failed offline IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Node=Offline
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock Nominal=Online
   |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
      |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
      '- Failed offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire Node=Offline
   '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
      |- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic

```

4. At this stage, you can perform the maintenance operations on the standby node, such as reboot OS.
5. After the maintenance work is completed, issue the **startprnode hostname** command for the standby node on the primary node. For example:

```
(P) #startprnode Zaire
(P) #lsrprnode
Name OpState RSCTVersion
Zaire Online 2.4.8.3
Baltic Online 2.4.8.3
```

6. Issue the **db2start** command on the standby instance:  
(S) \$db2start;db2 activate db sample
7. Issue the **lssam** command to observe the state of the resources. You are confirming that the resource of the instance on Zaire is online.

### ***Stop the primary node***

If you need to stop TSA on the primary node for the planned outage or maintenance, you can execute the following procedures:

1. Switch the roles. Have the standby (Zaire) take over the services so you can stop the stop the primary node (Baltic):

```
(S) $db2 takeover hadr on db sample
```

If the takeover completes successfully, the client should be able to access the new primary node (Zaire).

2. Issue the **db2stop force** command on the new standby instance:  
(S) \$db2stop force
3. Issue the **stopprnode** command as root user for the new standby node. You issue these commands on the old standby node:

```
(S) #stopprnode Baltic
(S) #lsrprnode
Name OpState RSCTVersion
Baltic Offline 2.4.8.3
Zaire Online 2.4.8.3
```

4. Issue the **lssam** command to observe the state of the resources.
5. At this stage, you can perform the maintenance operations on the new standby node, such as reboot OS.
6. Issue the **startprnode** command for the new standby node on the new primary node:

```
(P) #startprnode Baltic
(P) #lsrprnode
Name OpState RSCTVersion
Baltic Online 2.4.8.3
Zaire Online 2.4.8.3
```

7. Issue the **db2start** command on the new standby instance:

```
(S) $db2start;db2 activate db sample
```

8. Issue the **lssam** command to observe the state of the resources and confirm the resource of instance on Baltic is online.

9. Have the old primary node (Baltic) take over the services again:

```
(P) $db2 takeover hadr on db sample
```

After the takeover completes successfully, check that the client can access the primary node without any problems.

## 12.2.4 Unplanned outages

The purpose of having HA systems is to minimize the system down time caused by unplanned outages such as power failure, network failure, DB2 failure, and so on. In this section we simulate DB2 system failure, system crash, and network failure to verify our setup, as well as showing the transition states of the takeover process that TSA performed.

### DB2 instance failure

We simulate the DB2 instance failure on the primary node by stopping the DB2 processes.

#### *Stopping the DB2 instance on the primary*

We issue the **db2\_ki11** command on the primary instance to stop the DB2 processes. Now all DB2 clients cannot connect to the database.

We then use the **lssam** command to examine the resources. The HADR resource and the DB2 resource on the primary node are in the *Pending Online* state as shown in Example 12-14.

*Example 12-14 lssam output after stopping DB2*

---

```
(P)(S) #lssam
Pending online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
    '- Pending online IBM.Application:db2_db2inst1_Baltic_0-rs
        '- Pending online
IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
```

```

'- Online IBM.Application:db2_db2inst1_Zaire_0-rs
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
  | '- Pending online
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
  | '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
  | '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

---

TSA restarts the DB2 instance on the same node automatically. TSA does not execute takeover because the failed DB2 instance is recovered. All DB2 clients can connect to the database again.

### ***Preventing restart of DB2 instance on primary***

In this test, we rename the DB2 start executable file so TSA cannot restart the DB2 instance. We issue the following commands on the primary node to stop and break DB2 instance:

```
(P) $mv ./sqllib/adm/db2star2 db2star2.mv ; db2_kill
```

The output of the `1ssam` command shows that the HADR resource and the DB2 resource on the primary node are in the *Pending Online* state (Example 12-15). Now all DB2 clients cannot connect the database.

#### *Example 12-15 1ssam output after executing db2\_kill*

```

(P)(S) #1ssam
Pending online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Pending online IBM.Application:db2_db2inst1_Baltic_0-rs
  | '- Pending online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
  | '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
  | |- Pending online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
  | | '- Offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  | '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
  | |- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
  | | '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

---

TSA tries to start the DB2 instance but fails because the DB2 executable has been renamed. TSA then executes a failover. The virtual IP address is moved to the standby node, and the takeover operation causes the standby database to assume the primary role.

Shortly after the failover, the HADR resource group (db2\_db2inst1\_db2inst1\_SAMPLE-rg) is successfully placed in the *online* state. The resource on the old primary node is still in the *Pending Online* state as shown in Example 12-16.

*Example 12-16 Issam output after the takeover*

---

```
(P)(S) #lssam
Pending online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Pending online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Pending online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Failed offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
    '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
    '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

TSA continues trying to bring the DB2 instance resource online on what is now the old primary node. The timeout occurs 4 to 5 minutes afterwards, and this *Pending Online* state is changed to the *Failed Offline* state. See Example 12-17.

*Example 12-17 Issam output after timeout*

---

```
(P)(S) #lssam
Failed offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Failed offline IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Failed offline
      IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock
Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Failed offline
      IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
    '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
    '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---



To recover from this scenario, we rename the db2start executable to its original name:

```
(P) $mv ./db2star2.mv ./sql1lib/adm/db2star2
```

We have to reset the HADR and DB2 resources on the old primary node to get rid of the Failed Offline flag. This is done by issuing the **resetrsrc** commands (with root authority) in the given order on either the standby or the primary nodes:

```
resetrsrc -s 'Name like "instance resource name on old primary" &&  
NodeNameList={"old primary node name"}' IBM.Application
```

```
resetrsrc -s 'Name like "HADR resource name" && NodeNameList=  
{"old primary node name"}' IBM.Application
```

Example 12-18 shows the commands that we chose to run.

---

*Example 12-18 Reset the resources*

---

```
(P)(S) #resetrsrc -s 'Name like "db2_db2inst1_Baltic_0-rs" &&  
NodeNameList={"Baltic"}' IBM.Application
```

```
(P)(S) #resetrsrc -s 'Name like "db2_db2inst1_db2inst1_SAMPLE-rs" &&  
NodeNameList={"Baltic"}' IBM.Application
```

---

These commands result in the DB2 instance starting in the old primary. Reintegration occurs automatically, and the old primary database assumes the new standby role. After the system has reached peer state, the Lock status from the HADR resource group is removed. You can issue these procedures without impacting the activities taking place at the new primary database and DB2 clients.

## **Primary node crash and reintegration of old primary**

We discuss node failure in this section by simulating the node crash on the primary node. We can simulate this with power off or executing the **shutdown** command on the node.

### ***Node crash***

Figure 12-4 illustrates the cluster behavior in the event of a primary (service) node crash.

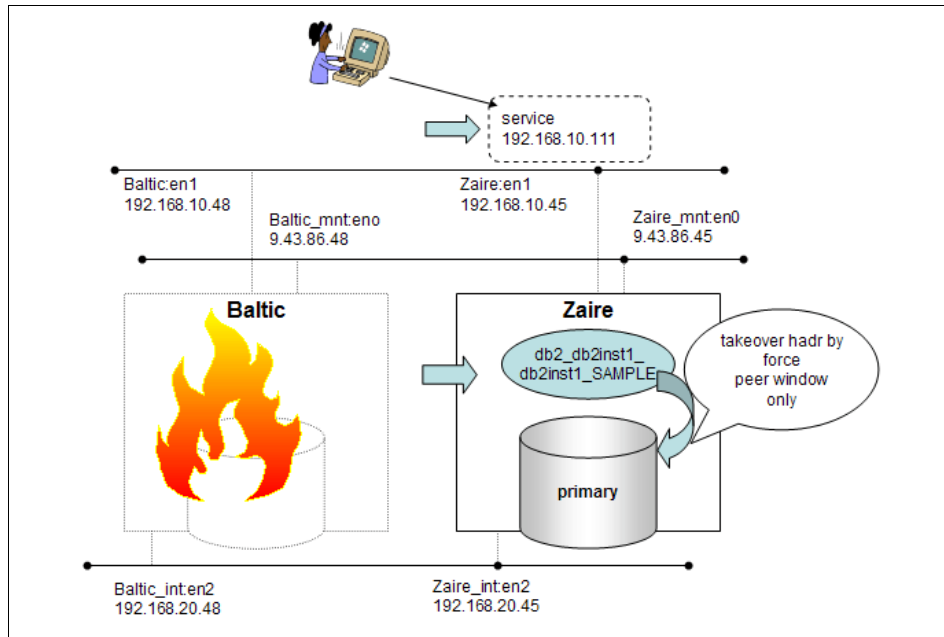


Figure 12-4 Automatic failover to standby node

When the primary node crashes, the clients cannot connect to the database. TSA detects the primary nodes outage and starts the failover process:

1. The resource group `db2_db2inst1_db2inst1_SAMPLE-rg` is acquired by the standby node (*Zaire*).
2. The standby node pings and acquires quorum.
3. The virtual IP address(192.168.10.111) is assigned to the en1 NIC on the standby node.
4. The Start script in the TSA, which is related to the resource group, is issued on the standby node. The script includes the TAKEOVER HADR command to change the role of the standby database to primary.

After the failover, the resources settle down to the states shown in Example 12-19:

- ▶ The HADR resource group `db2_db2inst1_db2inst1_SAMPLE-rg` is online on the new primary (*Zaire*). However, the HADR resource group is locked.
- ▶ The resources on the old primary node (*Baltic*) assume the *Failed Offline* state.

*Example 12-19 Issam output after the primary node crash*

---

```
(S) #lssam
Failed offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg
Control=StartInhibited Nominal=Online
    '- Failed offline IBM.Application:db2_db2inst1_Baltic_0-rs
        '- Failed offline
IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic Node=Offline
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
        '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Request=Lock
Nominal=Online
    |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
        |- Failed offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic Node=Offline
    '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
    '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
        |- Failed offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
Node=Offline
    '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

The HADR resource group is placed the *Lock* status after the failover. This lock indicates that the HADR databases are no longer in peer state. Therefore, *no* further actions are taken on this resource group if there are more failures.

Clients who address the service IP address succeed in connecting to the new primary database on the surviving node, which has the TSA resource group now.

### ***Reintegration***

Figure 12-5 shows the process required to reintegrate the old primary node into clusters:

1. Baltic is recovered from the crash.
2. As soon as the old primary node comes back up, the reintegration occurs automatically:
  - a. The DB2 instance is started automatically on the old primary (Baltic).
  - b. The old primary database is activated as a standby.
  - c. HADR replication resumes automatically. The old primary database automatically catches up the log records that are processed only on the new primary database during the time the old primary node is out of order.
  - d. As soon as the HADR system reaches *peer* state, the lock from the HADR resource group is removed. Reintegration of the old primary database is hereby complete.

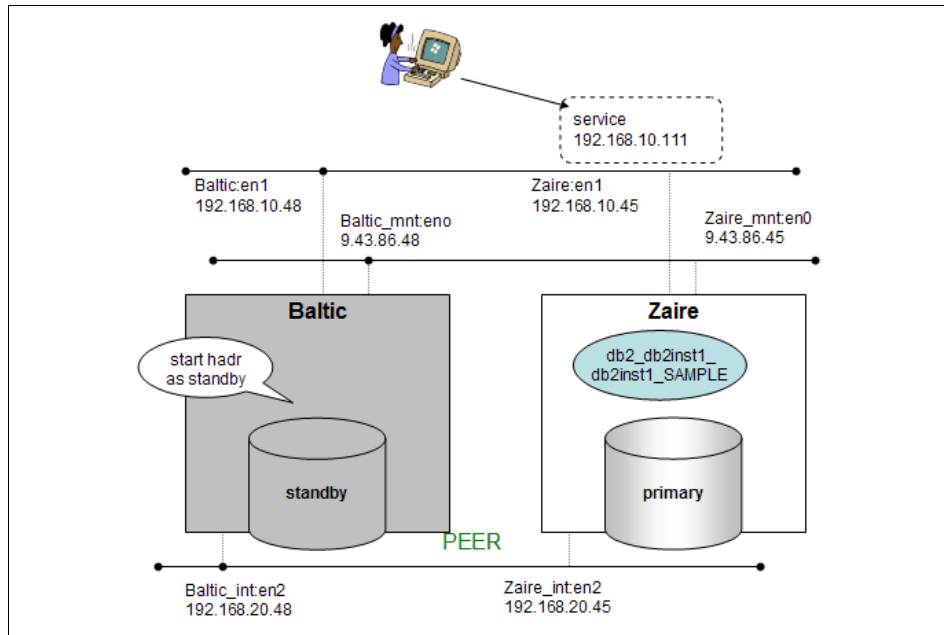


Figure 12-5 Reintegration of the old primary database

## Network failure

We discuss network failures in this section by simulating network interface malfunctions on the primary and the standby nodes.

### **Public network interface card failure on the primary node**

We unplug the en1 cable on the primary node to break the network, then use `lssam` to examine the state of the system resources. You can simulate the network failure using the `chdev` command.

```
(B) # chdev -l 'en1' -a state='detach'
```

The HADR resource (db2\_db2inst1\_db2inst1\_SAMPLE-rg) is in a Pending Offline State. See Example 12-20.

#### *Example 12-20 lssam output after network failure on primary*

```
(P)(S) #lssam
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
```

```

Pending offline IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg
Nominal=Online
  |- Pending offline IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
  |- Pending offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
  '- Offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs
  |- Failed offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
  '- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

---

TSA starts to execute a failover operation. We repeatedly issue the **lssam** command to examine the progress of the failover. The system eventually settles down to the following states:

- ▶ The standby HADR database assumes the primary role.
- ▶ The virtual IP address comes online on the standby node.
- ▶ The resource group of the old primary instance has the Failed Offline status as shown in Example 12-21.

*Example 12-21 lssam output after takeover*

---

```

(P)(S) #lssam
Failed offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg
Binding=Sacrificed Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_Baltic_0-rs
  '- Offline IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
  |- Offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
  '- Online
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
  |- Failed offline
IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

---

To recover from this state, we plug the en1 cable back into the old primary node. If you use **chdev** to simulate the failure, use this command again to change the state:

```
(B) # chdev -l 'en2' -a state='up'
```

Again, we repeatedly issue the `lssam` command to examine the progress of the failover. The system eventually settles down to the following states:

- ▶ The resource group of the old primary instance comes online.
- ▶ The old primary virtual IP resource remains Failed Offline.

See Example 12-22.

*Example 12-22 lssam output after recovering the network failure on the old primary*

---

```
(P)(S) #lssam
Online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Baltic_0-rs
    '- Online IBM.Application:db2_db2inst1_Baltic_0-rs:Baltic
Online IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_Zaire_0-rs
    '- Online IBM.Application:db2_db2inst1_Zaire_0-rs:Zaire
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
    |- Offline
IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Baltic
  '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
  '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    |- Failed offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
      '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

---

In this case, the virtual IP resource on the primary node remains in Failed Offline status. We need to reset the virtual IP resource on the old primary node to get rid of the Failed Offline flag. This is done by issuing the following command with root authority in the given order on either the standby or primary nodes.

```
resetrsrc -s 'Name like "Virtual IP resource name" &&
NodeNameList={"old primary node name"}' IBM.ServiceIP
```

For example:

```
(P) #resetrsrc -s 'Name like "db2ip_192_168_10_111-rs" &&
NodeNameList={"Baltic"}' IBM.ServiceIP
```

## 12.3 DB2 HADR configuration for automated failover on Linux

This section presents a test case that combines the new DB2 9.5 HA feature to automate failover of HADR databases on Linux.

## 12.3.1 Architecture

The example uses the following architecture, and assumes that the following minimum configuration is already in place:

- ▶ Two servers, lepus and mensa, running SUSE Linux Enterprise Server 10 SP1 x64
- ▶ One network interface card per server
- ▶ Common network access to a third gateway device as a tiebreaker
- ▶ DB2 9.5 fixpak1 with TSAMP installed
- ▶ HADR already set up on a database

Table 12-1 shows the naming conventions in use for this example:

*Table 12-1 DB2 HADR naming*

| <b>Server</b>                      | <b>lepus</b>                   | <b>mensa</b>                   |
|------------------------------------|--------------------------------|--------------------------------|
| IP                                 | 9.43.86.91                     | 9.43.86.90                     |
| local instance                     | db2inst2                       | db2inst2                       |
| local DB name                      | SAMPLE                         | SAMPLE                         |
| Remote node                        | mensa                          | lepus                          |
| Remote DB alias                    | SAMPMENS                       | SAMPLEPU                       |
| HADR database role                 | Primary                        | Standby                        |
| HADR_LOCAL_HOST                    | lepus                          | mensa                          |
| HADR_LOCAL_SVC                     | 55001                          | 55002                          |
| HADR_REMOTE_HOST                   | mensa                          | lepus                          |
| HADR_REMOTE_SVC                    | 55002                          | 55001                          |
| HADR_REMOTE_INST                   | db2inst2                       | db2inst2                       |
| HADR_TIMEOUT                       | 120                            | 120                            |
| HADR_SYNCMODE                      | SYNC                           | SYNC                           |
| HADR_PEER_WINDOW                   | 120                            | 120                            |
| Gateway tiebreaker                 | 9.43.85.1                      | 9.43.85.1                      |
| Virtual IP Address:<br>subnet mask | 9.43.86.252 :<br>255.255.252.0 | 9.43.86.252 :<br>255.255.252.0 |

**Note:** HADR prefers the actual service port number to the canonical service name for HADR\_LOCAL\_SVC and HADR\_REMOTE\_SVC. Local and remote host names in HADR\_LOCAL\_HOST and HADR\_REMOTE\_HOST should match the host short-names in /etc/hosts.

## 12.3.2 Configuration

In this section, we provide individual configuration steps, followed by the execution of the DB2 9.5 db2haicu utility to create a cluster. Steps and tasks involved in setting up a cluster integrated with DB2 9.5's HA feature are covered in detail in the DB2 manual, *Data Recovery and High Availability Guide and Reference*, SC23-5848-00, Chapter 3, in the section “Configuring a Clustered environment for high availability”.

### Before running db2haicu

Based on the provided architecture for our example, we carry out the following configuration steps before executing db2haicu:

1. Ensure that matching entries for DB2 instance communication and HADR ports exist in /etc/services files for both servers that are added as cluster nodes. Example 12-23 shows the entries for our configuration.

*Example 12-23 Matching DB2 service ports in /etc/services file on each node*

---

|               |       |
|---------------|-------|
| db2c_db2inst2 | 50002 |
| db2_hadr_1    | 55001 |
| db2_hadr_2    | 55002 |

---

2. Ensure that the matching host name entries exist in /etc/hosts files for both servers that are to be added as cluster nodes. Example 12-24 shows the entries for our configuration.

*Example 12-24 both hostname entries should be in both /etc/hosts files*

---

|            |                                    |
|------------|------------------------------------|
| 9.43.86.90 | mensa.itsosj.sanjose.ibm.com mensa |
| 9.43.86.91 | lepus.itsosj.sanjose.ibm.com lepus |

---

3. Run the ~/sqllib/db2profile script for the DB2 instance being used to create the cluster domain. This command has most likely already been added to the ~/.profile of the user ID by the **db2icrt** command.
4. Start the database manager on both servers using the **db2start** command.
5. Ensure that all DB2 High Availability Disaster Recovery (HADR) databases are started in their respective Primary and Standby database roles, and that all HADR Primary-Standby database pairs are in peer state.



Figure 12-6 shows DB2 being started and HADR in peer state in our systems.

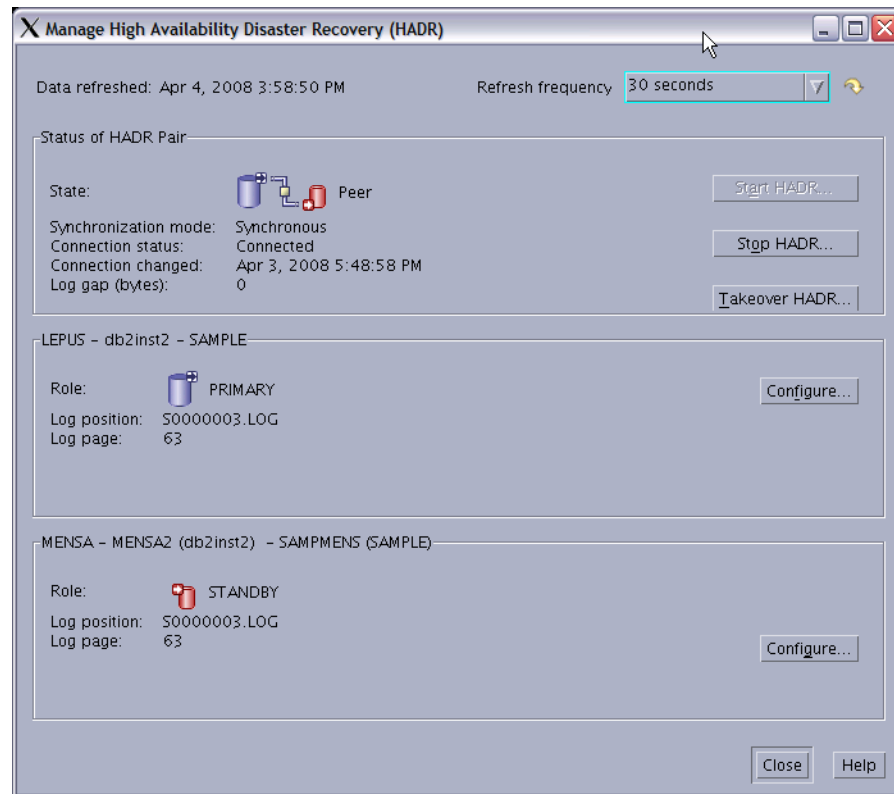


Figure 12-6 HADR connected and in peer state

- ▶ Set the new DB2 9.5 database configuration (db cfg) parameter HADR\_PEER\_WINDOW to a value greater than or equal to 120 [seconds] on both servers. Run the following command with instance owner ID:  

```
db2 update db cfg for sample using HADR_PEER_WINDOW 120
```
- ▶ To prepare each node for clustering:  
On both servers, run the **preprnode** command with user ID root:  

```
/usr/sbin/rsct/bin/preprnode lepus mensa
```

If this command is not run on both servers, “permission denied” errors occur when db2haicu is executed.

Example 12-25 shows the error message that is issued if preprnode is not run before db2haicu.

*Example 12-25 db2diag.log output with db2haicu issue preprnode*

---

```
2008-03-21-10.46.49.536683+600 E112086E1103          LEVEL: Error
PID      : 4891                                TID   : 47692343718912PROC : db2haicu
INSTANCE: db2inst1                            NODE  : 000
FUNCTION: DB2 Common, SQLHA APIs for DB2 HA Infrastructure, sqlhaCreateCluster,
probe:900
MESSAGE  : ECF=0x90000544=-1879046844=ECF_SQLHA_CREATE_CLUSTER_FAILED
          Create cluster failed
DATA #1 : String, 24 bytes
Error creating HA Domain
DATA #2 : String, 5 bytes
lepup
DATA #3 : unsigned integer, 4 bytes
3
DATA #4 : signed integer, 4 bytes
21
DATA #5 : String, 551 bytes
Line # : 8836---2632-044 The domain cannot be created due to the following
errors that were detected while harvesting information from the target nodes:
mensa: 2610-441 Permission is denied to access the resource class specified in
this command.
Network Identity UNAUTHENT requires 's' permission for the resource class
IBM.PeerDomain on node mensa.
```

---

Syntax for the preprnode command is covered by *RSCT for Linux Technical Reference*, SA22-7893-08 or RSCT Information Center:

[http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.doc/rsct\\_linux141/b15tr10837.html](http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.doc/rsct_linux141/b15tr10837.html)

At this stage, it is worth mentioning that if the environment is running on cloned virtual images (for example, VMware®) then the RSCT node identifier is not unique, which prevents creation of a cluster.

The node ID is a 64-bit number that is created when RSCT is installed. It is derived using a True Random Number Generator and is used to uniquely identify a node to the Resource Monitoring and Control (RMC) Subsystem. The node ID is maintained in the `/var/ct/cfg/ct_node_id` file. A backup copy is maintained in the `/etc/ct_node_id` file. Refer to the Information Center for details:

[http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.doc/rsct\\_aix5153/b15dia0417.html#chksman](http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.doc/rsct_aix5153/b15dia0417.html#chksman)

Example 12-26 shows the error message that is issued if db2haicu is run when RSCT node identifiers are not unique.

*Example 12-26 db2diag.log output for db2haicu issue RSCT node identifier*

---

```
2008-03-21-10.25.25.231360+600 E107866E943          LEVEL: Error
PID      : 29781                TID   : 47603185804288PROC : db2haicu
INSTANCE: db2inst1            NODE  : 000
FUNCTION: DB2 Common, SQLHA APIs for DB2 HA Infrastructure, sqlhaCreateCluster,
probe:900
MESSAGE  : ECF=0x90000544=-1879046844=ECF_SQLHA_CREATE_CLUSTER_FAILED
          Create cluster failed
DATA #1 : String, 24 bytes
Error creating HA Domain
DATA #2 : String, 5 bytes
lep us
DATA #3 : unsigned integer, 4 bytes
3
DATA #4 : signed integer, 4 bytes
21
DATA #5 : String, 391 bytes
Line #  : 8836---2632-044 The domain cannot be created due to the following errors
that were detected while harvesting information from the target nodes:
mensa: 2632-068 This node has the same internal identifier as lep us and cannot be
included in the domain definition.
```

---

This error can be corrected, if there is no cluster present, with the following command:

```
/usr/sbin/rsct/install/bin/recfgct
```

This command effectively re-initializes the RSCT base, and one feature is to give each node a unique identifier.

**Note:** If there is any cluster definition present when the **recfgct** command is executed, the effects are unpredictable, and symptoms might include immediate recycling of the operating system.

## **db2haicu configuration**

At this stage, the environment should be ready for the **db2haicu** command. It should be first executed on the HADR Standby node, and then on the HADR Primary node to successfully complete cluster definition with HADR integration.

Prior to DB2 9.5, the following two-stage process using **db2haicu** would consist of many manual tasks. Management of the cluster was driven by the cluster manager software rather than by any native DB2 commands.

When every prerequisite is met, the assumptions made by **db2haicu** allow use of the default response to almost each prompt. We use the defaults wherever possible, *emphasizing* places where we have made specific entries.

Example 12-27 shows the initial execution of db2haicu on our HADR Standby node after confirmation that HADR is running in peer state, as Standby.

*Example 12-27 db2haicu initial pass on HADR Standby node*

---

```
db2inst2@mensa:~> db2pd -d sample -hadr
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 4 days 16:54:37
```

HADR Information:

| Role    | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|---------|-------|----------|------------------|----------------------|
| Standby | Peer  | Sync     | 0                | 123949               |

| ConnectStatus | ConnectTime                          | Timeout |
|---------------|--------------------------------------|---------|
| Connected     | Tue Apr 8 09:03:34 2008 (1207670614) | 120     |

| PeerWindowEnd                        | PeerWindow |
|--------------------------------------|------------|
| Tue Apr 8 10:42:10 2008 (1207676530) | 120        |

| LocalHost | LocalService |
|-----------|--------------|
| MENSA     | 55002        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| LEPUS      | 55001         | db2inst2       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000004.LOG | 982       | 0x00000000026FEF5A |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000004.LOG | 942       | 0x00000000026D69C8 |

```
db2inst2@mensa:~> db2haicu
```

```
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst2. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

When you use db2haicu to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with db2haicu' in the DB2 Information Center. db2haicu is searching the current machine for an existing active cluster domain ...

db2haicu did not find a cluster domain on this machine. db2haicu will now query the system for information about cluster nodes to create a new cluster domain ...

db2haicu did not find a cluster domain on this machine. To continue configuring your clustered environment for high availability, you must create a cluster domain; otherwise, db2haicu will exit.

Create a domain and continue? [1]

1. Yes
2. No

Create a unique name for the new domain:

**lepumensa**

Nodes must now be added to the new domain.

How many cluster nodes will the domain lepumensa contain?

**2**

Enter the host name of a machine to add to the domain:

**mensa**

Enter the host name of a machine to add to the domain:

**lepus**

db2haicu can now create a new domain containing the 2 machines that you specified. If you choose not to create a domain now, db2haicu will exit.

Create the domain now? [1]

1. Yes
2. No

Creating domain lepumensa in the cluster ...

Creating domain lepumensa in the cluster was successful.

**Note:** If your server has some leftover cluster definitions, **db2haicu** might halt at this point and return you to the command line. You can either continue and re-issue the **db2haicu** command, which can skip over the next stage of creating a quorum device, or you can issue a **db2haicu -delete** command and then retry running **db2haicu** from scratch. You can add a quorum device later through the **db2haicu** maintenance mode menu item 6 “Create a new quorum device for the domain”. We recommend **db2haicu -delete**, because it assures you of a clean cluster definition.

You can now configure a quorum device for the domain. For more information, see the topic “Quorum devices” in the DB2 Information Center. If you do not configure a quorum device for the domain, then a human operator will have to manually intervene if subsets of machines in the cluster lose connectivity.

Configure a quorum device for the domain called lepumensa? [1]

1. Yes
2. No

The following is a list of supported quorum device types:

1. Network Quorum

Enter the number corresponding to the quorum device type to be used: [1]

Specify the network address of the quorum device:

### 9.43.85.1

Configuring quorum device for domain lepusmensa ...

Configuring quorum device for domain lepusmensa was successful.

The cluster manager found 2 network interface cards on the machines in the domain. You can use db2haicu to create networks for these network interface cards. For more information, see the topic 'Creating networks with db2haicu' in the DB2 Information Center.

Create networks for these network interface cards? [1]

1. Yes
2. No

Enter the name of the network for the network interface card: eth0 on cluster node:  
mensa.itsosj.sanjose.ibm.com

1. Create a new public network for this network interface card.
2. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node  
mensa.itsosj.sanjose.ibm.com to the network db2\_public\_network\_0? [1]

1. Yes
2. No

Adding network interface card eth0 on cluster node mensa.itsosj.sanjose.ibm.com to the  
network db2\_public\_network\_0 ...

Adding network interface card eth0 on cluster node mensa.itsosj.sanjose.ibm.com to the  
network db2\_public\_network\_0 was successful.

Enter the name of the network for the network interface card: eth0 on cluster node:  
lepus.itsosj.sanjose.ibm.com

1. db2\_public\_network\_0
2. Create a new public network for this network interface card.
3. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node  
lepus.itsosj.sanjose.ibm.com to the network db2\_public\_network\_0? [1]

1. Yes
2. No

Adding network interface card eth0 on cluster node lepus.itsosj.sanjose.ibm.com to the  
network db2\_public\_network\_0 ...

Adding network interface card eth0 on cluster node lepus.itsosj.sanjose.ibm.com to the  
network db2\_public\_network\_0 was successful.

Retrieving high availability configuration parameter for instance db2inst2 ...

Retrieving high availability configuration parameter for instance db2inst2 was  
successful.

Adding DB2 database partition 0 to the cluster ...

Adding DB2 database partition 0 to the cluster was successful.

Do you want to validate and automate HADR failover for the HADR database SAMPLE? [1]

1. Yes
2. No

**1**

```
Adding HADR database SAMPLE to the domain ...
The HADR database SAMPLE has been determined to be valid for high availability. However,
the database cannot be added to the cluster from this node because db2haicu detected
this node is the standby for the HADR database SAMPLE. Run db2haicu on the primary for
the HADR database SAMPLE to configure the database for automated failover.
All cluster configurations have been completed successfully. db2haicu exiting ...
db2inst2@mensa:~>
```

---

This brings us *almost* to the state of a working cluster. The HADR database pair has not yet been added to the cluster, because the database on the current cluster node is in HADR Standby role. This means that db2haicu must now be executed on the Primary node, to add the SAMPLE database for both nodes into the cluster.

On a clean system with no residual cluster definitions, the output of db2haicu in *startup mode* resembles Example 12-28. This shows db2haicu adding the HADR Primary database to the cluster definition, to pair with the HADR Standby database cluster definition. It also gives you the option to add in a virtual IP definition, as an alternative to using Automatic Client Redirection (ACR). We have chosen to use a virtual IP address in our test environment, with our remote client issuing **db2 catalog tcpip node** commands to point at that virtual IP address.

---

*Example 12-28 db2haicu on HADR Primary node - first run on a clean system*

---

```
lepus:~ # su - db2inst2
db2inst2@lepus:~> db2haicu
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst2. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

When you use db2haicu to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with db2haicu' in the DB2 Information Center. db2haicu is searching the current machine for an existing active cluster domain ...

db2haicu found a cluster domain called lepustumensa on this machine. The cluster configuration that follows will apply to this domain.

```

Retrieving high availability configuration parameter for instance db2inst2 ...
The cluster manager name configuration parameter (high availability configuration
parameter) is not set. For more information, see the topic "cluster_mgr - Cluster
manager name configuration parameter" in the DB2 Information Center. Do you want to set
the high availability configuration parameter?
The following are valid settings for the high availability configuration parameter:
  1.TSA
  2.Vendor
Enter a value for the high availability configuration parameter: [1]

Setting a high availability configuration parameter for instance db2inst2 to TSA.
Adding DB2 database partition 0 to the cluster ...
Adding DB2 database partition 0 to the cluster was successful.
Do you want to validate and automate HADR failover for the HADR database SAMPLE? [1]
  1. Yes
  2. No

Adding HADR database SAMPLE to the domain ...
Adding HADR database SAMPLE to the domain was successful.
Do you want to configure a virtual IP address for the HADR database SAMPLE? [1]
  1. Yes
  2. No

Enter the virtual IP address:
9.43.86.252
Enter the subnet mask for the virtual IP address 9.43.86.252: [255.255.255.0]
255.255.252.0
Select the network for the virtual IP 9.43.86.252:
  1. db2_public_network_0
Enter selection:
1
Adding virtual IP address 9.43.86.252 to the domain ...
Adding virtual IP address 9.43.86.252 to the domain was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
db2inst2@lepus:~>

```

---

And there we have it—a fully completed and integrated HADR cluster. We can see what cluster objects have been defined in the context of the TSA **1ssam** command in Example 12-29. This shows us, respectively:

- ▶ The resource group for the HADR database pair (SAMPLE on lepus, being HADR Primary, is shown “Online”; SAMPLE on mensa, being HADR Standby, is shown “Offline”).
- ▶ The Application definition of the virtual service IP address (again, lepus being HADR primary, is shown “Online”; mensa, being HADR Standby, is shown “Offline”).
- ▶ Two resource groups (one for each of the db2inst2 DB2 instances on lepus and mensa). Because both DB2 instances are technically running, both are shown as “Online”.



### Example 12-29 TSA view of cluster objects created by db2haicu

---

```
db2inst2@lepupus:~> lssam
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
    |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepupus
      '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepupus
      '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepupus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepupus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepupus_0-rs:lepupus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

## 12.3.3 Testing

In this section we go through the steps of a standard set of tests to prove that our cluster controlled by the DB2 9.5 HA feature is capable of handling common failure scenarios in an appropriate manner. These include unplanned failure scenarios of the DB2 instance on the HADR Primary node, and unplanned failure of the HADR Primary cluster node itself.

### Preparation

Our tests assume that our HADR database pair SAMPLE is active, connected, and in Peer state, and that our HADR Primary role is held by lepupus, and HADR Standby is mensa, as shown with **lssam** and **db2pd** commands in Example 12-30.

### Example 12-30 The assumed starting point for our testing scenarios

---

```
db2inst2@lepupus:~> lssam
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
    |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepupus
      '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepupus
      '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepupus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepupus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepupus_0-rs:lepupus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa

db2inst2@lepupus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:17:40

HADR Information:
Role   State           SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Peer           Sync       0                   0
```

|                                      |                                      |                    |
|--------------------------------------|--------------------------------------|--------------------|
| ConnectStatus                        | ConnectTime                          | Timeout            |
| Connected                            | Tue May 6 19:18:08 2008 (1210126688) | 120                |
| PeerWindowEnd                        | PeerWindow                           |                    |
| Tue May 6 19:37:38 2008 (1210127858) | 120                                  |                    |
| LocalHost                            | LocalService                         |                    |
| lepus                                | 55001                                |                    |
| RemoteHost                           | RemoteService                        | RemoteInstance     |
| mensa                                | 55002                                | db2inst2           |
| PrimaryFile                          | PrimaryPg                            | PrimaryLSN         |
| S0000005.LOG                         | 0                                    | 0x0000000002AF8000 |
| StandByFile                          | StandByPg                            | StandByLSN         |
| S0000005.LOG                         | 0                                    | 0x0000000002AF8000 |

---

## Monitoring scripts

During testing, we monitor the status of the cluster with similar scripts to those used in 11.4.2, “Testing topology response to common failures” on page 442. We use a single line shell script that repeatedly issues an `lssam` command, echoes an empty line, and sleeps for 5 seconds:

```
while : ;do lssam;echo “ “;sleep 5;done
```

We also monitor/test the remote DB2 connectivity of the current HADR Primary database with a simple looping script. Our example runs from a DB2 client also running on Linux, so it uses the following syntax:

```
while : ;do
db2 connect to testsamp user db2inst2 using passblah
db2 connect reset
echo “ “
sleep 5
done
```

The DB2 client script assumes the following catalog TCP/IP node and database alias definition executed on the remote client DB2 command line:

```
db2 catalog tcpip node testsamp remote 9.43.86.252 server 50002
db2 catalog db sample as testsamp at node testsamp
```

As shown, we use the Virtual IP address defined as part of the `db2haicu` cluster definition as the service IP address (that is, the address that external clients use to connect to the current DB2 HADR Primary database).

## Planned HADR takeover

A wonderful property of the DB2 9.5 HA feature is that appropriate cluster actions are integrated into DB2 native commands.

For HADR clusters controlled by the new HA feature, rather than needing to **su** to the root user and issue **chrg** or whichever cluster manager command normally required to make a change in the cluster, we simply issue **db2stop**, **db2start**, **stop hadr**, **start hadr**, **takeover hadr** commands as required. The cluster reacts appropriately, rather than fighting against these actions on the assumption that something is wrong with a resource. The DB2 instance user is also authorized to execute the **1ssam** command in order to monitor the current Online/Offline state of cluster resources. There is no requirement to **su** to the root user.

A more complete list of the DB2 native commands integrated with the cluster manager can be found in the DB2 9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.1uw.admin.ha.doc/doc/t0051380.html>

Our first test is to issue two unforced HADR takeovers. The first one moves the HADR Primary role from the SAMPLE database on lepus to the SAMPLE database on mensa, and the second one moves the HADR Primary role back again, to the SAMPLE database on lepus.

We start our looping **1ssam** script on one of the servers, then have the db2inst2 user on mensa (current HADR Standby) issue the following command:

```
db2 takeover hadr on db sample
```

The step-by-step results for our first **takeover** command are shown from Example 12-31 here to Example 12-36 on page 523

The first change is a *Request=Lock* state on the HADR resource group in Example 12-31.

### *Example 12-31 Unforced takeover - step 1 of 6*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

The next noticeable change is a Request=Move, and the virtual IP address is set *Offline* in Example 12-32.

*Example 12-32* *Unforced takeover - step 2 of 6*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Move Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

In Example 12-33, the HADR resource for the primary is set to *Pending Offline*.

*Example 12-33* *Unforced takeover - step 3 of 6*

---

```
Pending offline IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Move Nominal=Online
|- Pending offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Pending offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

The virtual IP address for mensa has been set to *Online*, and the HADR database resource for mensa is *Pending Online*, in Example 12-34.

*Example 12-34* *Unforced takeover - step 4 of 6*

---

```
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Move Nominal=Online
|- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

The HADR database resource for mensa is now set to *Online*. Apart from the *Request=Lock*, the DB2 HADR pair should now be available again for communication as shown in Example 12-35.

### Example 12-35 Unforced takeover - step 5 of 6

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

This is the point at which control of the DB2 command line returns to the administrator. The SAMPLE database on lepus now has the HADR Standby role in Example 12-36, confirmed with a follow-up **db2pd** command. With the SAMPLE database on mensa having the HADR Primary role, the Service IP address resource has also been assigned to mensa.

### Example 12-36 Unforced takeover - step 6 of 6

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

```
db2inst2@lepus:~> db2pd -d sample -hadr
```

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 02:21:49
```

```
HADR Information:
```

| Role         | State | SyncMode | HeartBeatsMissed | LogGapRunAvg (bytes) |
|--------------|-------|----------|------------------|----------------------|
| Standby Peer |       | Sync     | 0                | 177447               |

| ConnectStatus | ConnectTime                          | Timeout |
|---------------|--------------------------------------|---------|
| Connected     | Tue May 6 19:18:08 2008 (1210126688) | 120     |

| PeerWindowEnd                        | PeerWindow |
|--------------------------------------|------------|
| Tue May 6 23:13:21 2008 (1210140801) | 120        |

| LocalHost | LocalService |
|-----------|--------------|
| lepus     | 55001        |

| RemoteHost | RemoteService | RemoteInstance |
|------------|---------------|----------------|
| mensa      | 55002         | db2inst2       |

| PrimaryFile  | PrimaryPg | PrimaryLSN         |
|--------------|-----------|--------------------|
| S0000005.LOG | 43        | 0x0000000002B23522 |

| StandByFile  | StandByPg | StandByLSN         |
|--------------|-----------|--------------------|
| S0000005.LOG | 0         | 0x0000000002AF8000 |

---

The output from our looping remote DB2 client script in Example 12-37 shows that for a brief period during the cluster manager's resource switching actions, the Service IP address is pointing to the HADR Standby database, then is unavailable, and then connectivity returns to normal.

*Example 12-37 Brief outage of remote DB2 connectivity during takeover*

---

Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST2
Local database alias = TESTSAMP
```

DB20000I The SQL command completed successfully.

SQL1776N The command cannot be issued on an HADR standby database. Reason code = "1".

SQL1024N A database connection does not exist. SQLSTATE=08003

SQL1776N The command cannot be issued on an HADR standby database. Reason code = "1".

SQL1024N A database connection does not exist. SQLSTATE=08003

SQL30081N A communication error has been detected. Communication protocol being used: "TCP/IP". Communication API being used: "SOCKETS". Location where the error was detected: "9.43.86.145.252". Communication function detecting the error: "connect". Protocol specific error code(s): "113", "\*\*", "\*". SQLSTATE=08001

SQL1024N A database connection does not exist. SQLSTATE=08003

Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST2
Local database alias = TESTSAMP
```

DB20000I The SQL command completed successfully.

---

The only reason we are even seeing the SQL1776N messages in Example 12-37 is that for this particular test environment, we have not configured DB2 Automatic Client Reroute (ACR) between the two HADR databases. This provides a useful contrast with the output from testing that we perform in 11.4, “Automating HADR takeover with TSA on Linux” on page 405, where we configure ACR in addition to the Virtual IP resource. We can therefore recommend ACR in addition to a cluster managed virtual IP address, as a useful means of avoiding remote DB2 connections pointing at an HADR Standby database.

We now return HADR Primary role back to the SAMPLE database on lepus, with the following command issued by db2inst2 on lepus:

```
db2 takeover hadr on db sample
```

Because the process of unforced HADR takeover from mensa back to lepus shows very much the same output in Issam, only with the other resources in a given resource group pair doing the *Pending Offline - Offline - Pending Online - Online* status changes, we summarize the outcome in Example 12-38.

### Example 12-38 Output from db2pd and lssam after takeover back to lepus

---

```
db2inst2@lepus:~> db2 takeover hadr on db sample
DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.
db2inst2@lepus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 02:25:49

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary Peer              Sync           0                  0

ConnectStatus ConnectTime                               Timeout
Connected    Tue May 6 19:18:08 2008 (1210126688) 120

PeerWindowEnd PeerWindow
Tue May 6 21:45:51 2008 (1210135551) 120

LocalHost      LocalService
lepus          55001

RemoteHost     RemoteService RemoteInstance
mensa          55002          db2inst2

PrimaryFile PrimaryPg PrimaryLSN
S0000005.LOG 43      0x0000000002B23522

StandByFile StandByPg StandByLSN
S0000005.LOG 43      0x0000000002B23522
db2inst2@lepus:~>

Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

## Unplanned failure of the HADR primary DB2 instance

Because the DB2 9.5 HA feature has integrated the **db2stop** command into cluster management, we have to use a technique that effectively stops the DB2 instance outside the knowledge of the cluster manager.

The **db2\_ki11** command is an excellent way to achieve this. To test, we first start our looping lssam monitor script on either one of the cluster nodes, and our DB2 connect script on the remote client, then have the DB2 instance user perform the **db2\_ki11** command on our HADR primary node lepus, and capture the results.

The `db2_kill` command works as expected in Example 12-39.

*Example 12-39 db2\_kill performs its function*

---

```
db2inst2@lepus:~> db2_kill
ipclean: Removing DB2 engine and client's IPC resources for db2inst2.
```

---

The looping `Issam` script output from Example 12-40 to shows that the first preferred action of the cluster manager in this situation, is to simply attempt to restart the DB2 instance which failed. This exemplifies the importance of ensuring that if you really want to make DB2 resources unavailable for maintenance purposes, you should use commands that the cluster manager is aware of, and does not automatically attempt to override your actions.

In Example 12-40 we see the first noticeable change from nominal resource states, to the cluster manager noticing that something is wrong, placing the DB2 instance resource group for `lepus` into a *Pending Online* state.

*Example 12-40 Unplanned failure - Issam output 1 of 3*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
.
.
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Pending online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Pending online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Example 12-41 shows the momentary lapse in the cluster manager's determination of the resource state for the HADR Primary database. All resource states first appear nominal, but this is picked up in the next iteration, where the HADR Primary database resource on `lepus` is shown with a state of *Unknown*. This is where the HADR Primary database has not yet been activated, and DB2 needs to perform crash recovery.



### Example 12-41 Unplanned failure - Issam output 2 of 3

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Pending online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Pending online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Example 12-42 shows the final range of actions taken by the cluster manager. The HADR database resource for lepus is set to *Pending Online* while DB2 performs crash recovery, then all resource states are back to nominal. DB2 connectivity is working again after the HADR Primary database is activated.

### Example 12-42 Unplanned failure - Issam output 3 of 3

---

```
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
  '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Our output from the looping DB2 connect script shows that the DB2 connectivity is lost, then returns. Because we have not enabled ACR, this also indicates to us that the HADR database role and the Virtual IP address are never switched by the cluster manager for our test scenario. The only action the cluster manager takes is to simply restart the failed DB2 instance and to activate the database as HADR Primary. This matches the behavior of the TSA cluster we test in 11.4, “Automating HADR takeover with TSA on Linux” on page 405.

When the cluster has returned all resources to their nominal state, we issue a **db2pd** command in Example 12-43, which confirms that the SAMPLE database on lepus has retained the HADR Primary role.

*Example 12-43 Return of the HADR Primary*

---

```
db2inst2@lepus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:58

HADR Information:
Role      State          SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Peer          Sync          0                   0

ConnectStatus ConnectTime                               Timeout
Connected    Wed May  7 17:41:26 2008 (1210207286) 120

PeerWindowEnd PeerWindow
Wed May  7 17:43:57 2008 (1210207437) 120

LocalHost      LocalService
lepus          55001

RemoteHost     RemoteService RemoteInstance
mensa          55002         db2inst2

PrimaryFile PrimaryPg PrimaryLSN
S0000006.LOG 43      0x0000000002F0B522

StandByFile StandByPg StandByLSN
S0000006.LOG 43      0x0000000002F0B522
```

---

### Unplanned failure of the primary cluster node

Our final test in this environment is to completely remove the primary node, and see what actions are taken by the cluster manager on the remaining node. Effectively, the Primary node is in a suspended state, unable to be restarted by the cluster manager on either node. We manually restart this node ourselves and see how easily it can be re-integrated back into the cluster, and into the HADR database pair.

After starting the remote DB2 connect script on the client, and the lssam script on the HADR standby node mensa, our next action is to stop the Primary cluster node in an unplanned manner. The most generic way to achieve this is with a **shutdown -h now** command issued by the root user.

The output from our `Issam` script in Example 12-44 and Example 12-45 shows the range of actions from when the cluster manager notices that DB2 is unavailable, until it assigns the HADR Primary role to the SAMPLE database on our mensa node. At this point, our lepus node is completely powered off.

Example 12-44 shows the cluster manager setting the resources for our lepus node to *Offline* and *Failed Offline*. At this stage, to preserve integrity while it decides what appropriate action to take, the cluster manager also sets the resource group state for the mensa DB2 instance `db2inst2` to *Offline*, although it detects that the actual DB2 instance resource is still *Online*.

*Example 12-44 Something is not right here - but no need to panic.*

---

```

Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
   |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
   '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mena
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
   |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
   '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mena
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
   '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
   '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mena
.
Offline IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
   |- Failed offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus Node=Offline
   '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mena
'- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs
   |- Failed offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus Node=Offline
   '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mena
Offline IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs
   '- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs:lepus Node=Offline
Offline IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
   '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mena

```

---

Example 12-45 shows the cluster manager progressing from noticing that the Primary node has gone, and is not going to be coming back any time soon (*Failed Offline*). It then sets the HADR Resource group to *Pending Online*, and then to *Online* as it successfully performs an HADR forced takeover, giving the HADR Primary role to the SAMPLE database on mensa. Of particular interest in a node failure scenario is the `Control=StartInhibited` flag issued against the DB2 instance resource group for lepus. This protects the integrity of the cluster in case lepus decides to go online again. Before lepus allows any external connectivity, it needs to perform appropriate cluster actions to satisfy requirements to overcome the `Control=StartInhibited` flag.

### Example 12-45 Something very bad - but cluster manager can handle it

---

```
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Failed offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus Node=Offline
  '- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
      |- Failed offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus Node=Offline
      '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Failed offline IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Control=StartInhibited Nominal=Online
'- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs
  '- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs:lepus Node=Offline
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Failed offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus Node=Offline
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
      |- Failed offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus Node=Offline
      '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Failed offline IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Control=StartInhibited Nominal=Online
'- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs
  '- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs:lepus Node=Offline
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

The output from our DB2 client connect script in Example 12-46 shows no point in time where DB2 displays lost connectivity. A forced takeover and Virtual IP switch managed by our cluster manager are effective and rapid.

### Example 12-46 DB2 Connect™ output - boring, nothing to see here

---

```
db2inst1@itso:~/script> ./conntest.sh

Database Connection Information

Database server          = DB2/LINUX8664 9.5.0
SQL authorization ID    = DB2INST2
Local database alias    = TESTSAMP

DB20000I The SQL command completed successfully.

...
```

---

The second part of our test consists of manually restoring power to our lepus node, and continuing to capture the results as it automatically re-integrates into the cluster.

Example 12-47 shows the transition from when the cluster manager detects that the lepus node is back and available for cluster activity.

*Example 12-47 lepus is alive - again - almost.*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Failed offline IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Control=StartInhibited Nominal=Online
'- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs
  '- Offline IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Offline IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Pending online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Pending online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Example 12-48 shows the next transition with our lepus node now fully re-integrated back into the cluster with a DB2 HADR Standby role. The DB2 instance resource group for lepus is set to *Online*, and the DB2 HADR database resource for lepus is changed from *Unknown* to *Offline*, meaning it has been successfully started as an HADR Standby database.

*Example 12-48 lepus is alive, and HADR is standing by.*

---

```
Offline IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Unknown IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lepus_0-rs
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
```

```
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
'- Online IBM.Application:db2_db2inst2_mensa_0-rs:mena
```

---

For DB2 connect output, we refer you again to Example 12-46 on page 530. DB2 connectivity is never lost. Example 12-49 shows that DB2 HADR is Connected and in Peer state, and that the SAMPLE database has the Standby role on lepus.

*Example 12-49 Sometimes you just have to Stand by.*

---

```
db2inst2@lepus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:51

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Standby Peer              Sync      0                  0

ConnectStatus ConnectTime              Timeout
Connected     Wed May 7 19:35:16 2008 (1210214116) 120

PeerWindowEnd PeerWindow
Wed May 7 15:17:55 2008 (1210198675) 120

LocalHost      LocalService
lepus          55001

RemoteHost     RemoteService RemoteInstance
mena          55002         db2inst2

PrimaryFile PrimaryPg PrimaryLSN
S0000006.LOG 43      0x0000000002F0B522

StandByFile StandByPg StandByLSN
S0000006.LOG 43      0x0000000002F0B522
```

---

Our final step to bring our cluster back to its initial state, is to issue an unforced HADR takeover command from lepus - results are as expected, with the db2pd output shown in Example 12-50. Output from our remote DB2 Connect script during this process mirrored Example 12-37 on page 524 exactly.

*Example 12-50 Sometimes you have to take on a Primary role.*

---

```
db2inst2@lepus:~> db2 takeover hadr on db sample
DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.
db2inst2@lepus:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 01:19:18

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary Peer              Sync      0                  84890

ConnectStatus ConnectTime              Timeout
Connected     Wed May 7 19:35:16 2008 (1210214116) 120

PeerWindowEnd PeerWindow
Wed May 7 20:56:19 2008 (1210218979) 120

LocalHost      LocalService
lepus          55001
```

|              |           |                    |                |
|--------------|-----------|--------------------|----------------|
| RemoteHost   |           | RemoteService      | RemoteInstance |
| mensa        |           | 55002              | db2inst2       |
| PrimaryFile  | PrimaryPg | PrimaryLSN         |                |
| S0000006.LOG | 84        | 0x0000000002F34C57 |                |
| StandByFile  | StandByPg | StandByLSN         |                |
| S0000006.LOG | 43        | 0x0000000002F0B522 |                |

---

This concludes our testing of HADR integrated into a managed cluster with the DB2 9.5 HA feature.

## 12.3.4 Administration

This is an extremely brief overview of how to perform maintenance activities on DB2 resources without having the cluster manager attempting to restart them while you require them to be offline.

We have mentioned in “Planned HADR takeover” on page 521 how DB2 commands are now integrated into the cluster manager. This makes administration and maintenance of DB2 objects extremely intuitive.

For our examples here, we use the same test environment and initial cluster state as for 12.3.3, “Testing” on page 519.

Our first example of performing maintenance is to simply issue a **db2stop** command on the DB2 instance with the HADR standby database. We can once again measure the results using our looping *Issam* script from “Monitoring scripts” on page 520.

Example 12-51 shows the output from the **db2stop** command. As you can see, we first need to issue a DB2 **deactivate database** command against an HADR standby database, otherwise DB2 complains.

### *Example 12-51 db2stop on the HADR Standby instance*

---

```

db2inst2@mensa:~> db2stop
05/07/2008 17:00:20    0    0    SQL1025N  The database manager was not stopped because databases are
still active.
SQL1025N  The database manager was not stopped because databases are still active.
db2inst2@mensa:~> db2 deactivate db sample
DB20000I  The DEACTIVATE DATABASE command completed successfully.
db2inst2@mensa:~> db2stop
05/07/2008 17:01:10    0    0    SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.

```

---

Example 12-52 shows the *Issam* output. After the cluster manager accepts the request, the resource group for the DB2 instance is set to *Offline*, and the HADR Standby database resource and virtual IP resource remain as *Offline*. This state does not change until you decide to manually restart the DB2 instance.

### Example 12-52 DB2 stopped with DB2 HA feature

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Offline IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Offline IBM.Application:db2_db2inst2_mensa_0-rs
    '- Offline IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Example 12-53 shows that we are manually restarting the DB2 instance, and the output of a **db2pd** command to confirm that the HADR Standby database is automatically activated. We performed the **db2pd** test primarily because it is difficult to tell simply from output of the **Issam** command whether the HADR resource is in standby or is really offline.

### Example 12-53 DB2 starts and the HADR Standby is activated automatically

---

```
db2inst2@mensa:~> db2start
05/07/2008 17:17:54 0 0 SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
db2inst2@mensa:~> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:06

HADR Information:
Role State SyncMode HeartBeatsMissed LogGapRunAvg (bytes)
Standby Peer Sync 0 0

ConnectStatus ConnectTime Timeout
Connected Wed May 7 17:18:16 2008 (1210205896) 120

PeerWindowEnd PeerWindow
Null (0) 120

LocalHost LocalService
mensa 55002

RemoteHost RemoteService RemoteInstance
lepus 55001 db2inst2

PrimaryFile PrimaryPg PrimaryLSN
S0000006.LOG 88 0x000000002F38807
```



Example 12-54 shows the `Issam` output with a very simple transition between the DB2 instance being stopped (*Offline*), and us restarting it to be automatically reintegrated into the cluster (*Online*). Note that as far as the cluster manager is telling us, there is no change in state for the HADR Standby database resource, although we know that it has to internally keep track of the exact state of an HADR Standby database for the cluster to function correctly. The *Online - Offline* state limitation is an appropriate peculiarity of cluster managers, and resource dependencies.

*Example 12-54 Reintegration is smooth with the DB2 HA feature*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
|  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
|  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
|- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
|  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
|  |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_lepus_0-rs
|  |- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Offline IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst2_mensa_0-rs
|  |- Offline IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. . .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
|  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
|  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
|- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
|  |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
|  |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_lepus_0-rs
|  |- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_mensa_0-rs
|  |- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

If a planned outage is required on the DB2 instance where the HADR Primary database is currently active, even without a cluster, the appropriate action for a DB2 administrator to follow is to first issue an unforced takeover to switch the HADR Primary role across to the database on the other node. After a successful unforced takeover, the database can be deactivated, and the instance stopped, in order to perform administration or maintenance activities.

We have seen in “Unplanned failure of the HADR primary DB2 instance” on page 525 how the cluster manager reacts to a `db2_kill` command issued outside its control, but for our next example, we attempt to issue a `db2stop` against the DB2 instance with the HADR Primary database, and see what eventuates.

Example 12-55 is the output from our attempt to stop a DB2 HADR primary database and instance.

*Example 12-55 Stopping an HADR primary resource*

---

```
db2inst2@lep us:~> db2 list applications
SQL1611W No data was returned by Database System Monitor.
db2inst2@lep us:~> db2 deactivate db sample
DB20000I The DEACTIVATE DATABASE command completed successfully.
db2inst2@lep us:~> db2stop
05/07/2008 22:09:29 0 0 SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
```

---

Example 12-56 to Example 12-58 illustrate what we suspected might happen. The cluster takes action to restart the DB2 instance and reactivate the HADR Primary database.

*Example 12-56 Stopping HADR primary - part 1 of 3*

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
|   |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lep us
|   '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
|   |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lep us
|   '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lep us_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_lep us_0-rs
|   '- Online IBM.Application:db2_db2inst2_lep us_0-rs:lep us
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
|   '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. . .
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
|   |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lep us
|   '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
'- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
|   |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lep us
|   '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst2_lep us_0-rg Nominal=Online
'- Pending online IBM.Application:db2_db2inst2_lep us_0-rs
|   '- Pending online IBM.Application:db2_db2inst2_lep us_0-rs:lep us
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst2_mensa_0-rs
|   '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

Example 12-57 shows the progression from the DB2 instance being restarted and set back to *Online* state, to where the HADR database resource for lepus has been optimistically set to *Pending Online*.

**Example 12-57 Stopping HADR primary - part 2 of 3**

---

```
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
. .
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

In Example 12-58, we see that the cluster reports the HADR database resource group and resource for lepus have been set to *Online*, and all is back to nominal.

**Example 12-58 Stopping HADR primary - part 3 of 3**

---

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mensa
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mensa
Online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_lepus_0-rs
    '- Online IBM.Application:db2_db2inst2_lepus_0-rs:lepus
Online IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst2_mensa_0-rs
    '- Online IBM.Application:db2_db2inst2_mensa_0-rs:mensa
```

---

## 12.4 DB2 with shared storage HA on AIX

In this section, we show you how to configure a DB2 shared storage HA cluster using the db2haicu utility on the AIX platform.

### 12.4.1 Architecture

DB2 with shared storage HA is the most typical way of creating high availability clusters with DB2. This type of cluster has a shared storage that can be owned by every server in the cluster. During normal operation, only one server has the ownership of the shared disk. When the cluster software detects a failure on the owner node, cluster software fails the resources over to the other server to continue the DB2 services.

Figure 12-7 shows the resource mapping in normal operation.

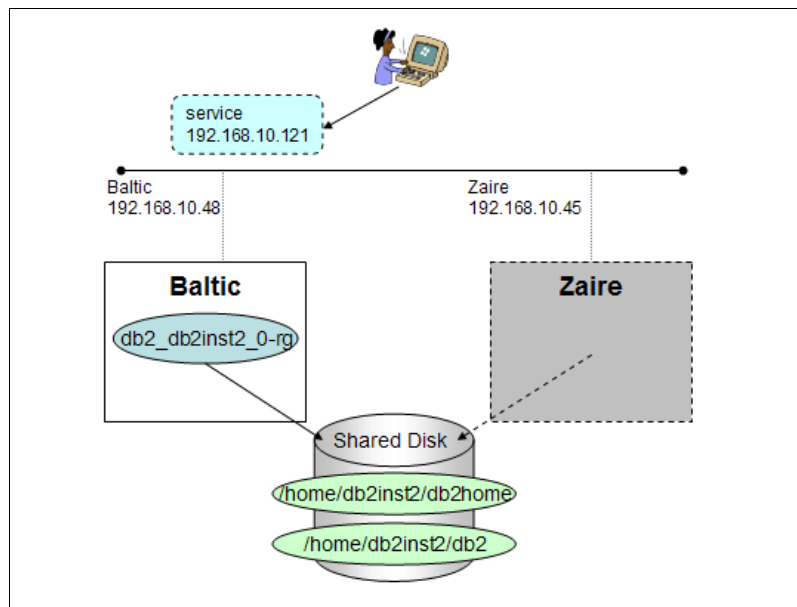


Figure 12-7 Resources state during normal operation

Figure 12-8 shows the moved resources after takeover.

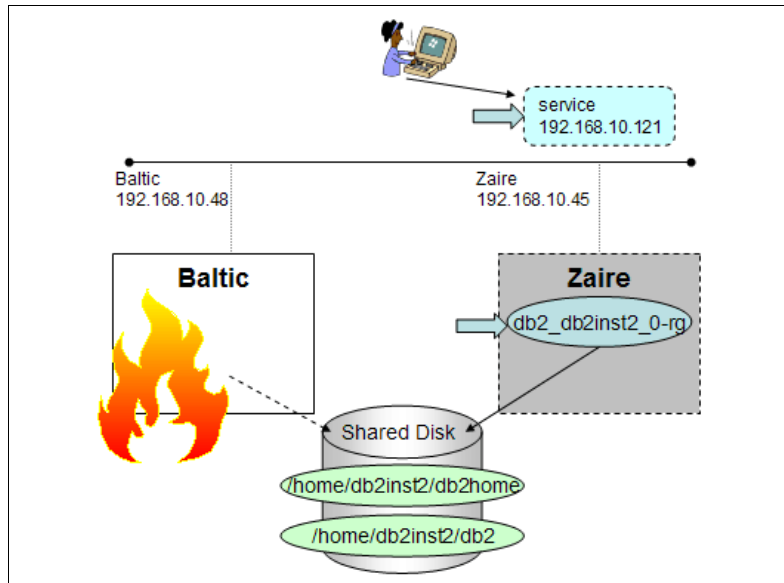


Figure 12-8 Resources when Baltic was failed

We conduct the HA scenario in this section under the following lab environment:

- ▶ Two servers, Baltic and Zaire, are running on AIX5.3 TL7 SP3-0811.
- ▶ External storage DS4700 is connected through a SAN switch.
- ▶ Three network interface cards. The requirement for the network interface card is one or more.
- ▶ Each server can reach the third machine as a tiebreaker.
- ▶ DB2 Version 9.5 FP1 with TSAMP Version 2.2 is installed.

Table 12-2 shows the resources and naming convention used in this scenario.

Table 12-2 Resources and naming convention

| Server                    | Baltic                 | Zaire         |
|---------------------------|------------------------|---------------|
| IP address                | 192.168.10.48          | 192.168.10.45 |
| Local instance            | db2inst2               |               |
| Instance home file system | /home/db2inst2/db2home |               |
| Database data file system | /home/db2inst2/db2     |               |
| Service IP address        | 192.168.10.121         |               |
| Database name             | SAMPLE                 |               |
| HA role                   | Active                 | Passive       |

## 12.4.2 Configuration

In this section we describe a step-by-step procedure for configuring a cluster domain using db2haicu utility with XML file. For the setup example using db2haicu interactive mode, refer to 12.3, “DB2 HADR configuration for automated failover on Linux” on page 508.

The procedure to configure a highly available DB2 environment with shared storage consists of the following steps:

1. Check prerequisites.
2. Check network interface.
3. Set up the storage for sharing.
4. Create DB2 instance.
5. Cluster preparation.
6. System clock synchronization.
7. Create configuration XML file for db2haicu.
8. Cluster domain setup by db2haicu with XML file.
9. Assess the created cluster domain.

We describe each step in detail in the following sections.

### Checking prerequisites

The first configuration step is checking prerequisite of db2haicu. The details are described in the following sections:

- ▶ 12.1, “db2haicu” on page 478
- ▶ 12.2, “DB2 HADR configuration for automatic failover with on AIX” on page 483

### Checking the network interface

db2haicu uses two types of networks, a public network and a private network. In the DB2 shared storage HA scenario, we need at least one public network to provide a service IP address.

We assign the following static IP addresses to the en1 adapters on the owner and failover nodes:

```
Owner node (Baltic)
  en1: 192.168.10.45 (255.255.252.0)
Failover node (Zaire)
  en1: 192.168.10.48 (255.255.252.0)
```

Make sure that /etc/hosts file contains entries of the owner node and the failover node names. All cluster nodes should have the same entries in the /etc/hosts file. In addition, all of them must have static IP addresses.

In our example, we have these entries in the `/etc/hosts` file of both servers:

```
192.168.10.45    Zaire
192.168.10.48    Baltic
```

Two servers have to be able to communicate to each other through a public network. We confirm it by a ping command. Execute the following commands on both nodes and make sure that they complete successfully:

```
(0) (F) $ ping Baltic
(0) (F) $ ping Zaire
```

## Setting up the storage for sharing

Before proceeding with the steps to set up the storage for share between two nodes, configure the storage and create the volume groups. Each node of the cluster domain must be able to access the shared volume groups. The owner node keeps the ownership of the shared volume groups during the normal operation. The failover node takes over the ownership and varies on the volume groups automatically in case of a takeover. Therefore, make sure that both servers can vary on the volume groups. We explain how to set up the storage on “Setting up the storage for sharing” on page 249.

## Creating a DB2 instance

Use the following steps to create a DB2 instance on a shared disk:

1. Check the user ID and group ID of the DB2 instance owner on the nodes.

If there is no DB2 instance owner user, you have to create the DB2 instance owner user and group first. The DB2 instance owner should have the same user ID and group ID on all nodes in the cluster domain. We recommend that the DB2 instance owner have the same password on all cluster nodes.

Example 12-59 shows that the user ID of DB2 instance owner `db2inst2` is 205 and its group ID is 102. We ensure that they are the same in every node.

*Example 12-59 Checking sample of user ID and group ID*

---

```
(0) (F) root@Baltic # lsuser -a ID pgrp db2inst2
db2inst2 ID=205 pgrp=db2iadm1
(0) (F) root@Baltic # lsgroup -a ID db2iadm1
db2iadm1 ID=102
```

---

2. Create the DB2 instance.

We use the `db2icrt` command to create the DB2 instance. The command is in the instance directory on the DB2 install path.

```
(0) # <DB2 install path>/instance/db2icrt -u <fenced user name>
<instance owner name>
```

Example 12-60 and Example 12-61 illustrate how to create the DB2 instance and configure TCP/IP communication.

*Example 12-60 DB2 instance creation*

---

```
(0) root@Baltic # cd /opt/IBM/db2/V9.5/instance
(0) root@Baltic # ./db2icrt -u db2fenc1 db2inst2
DBI1070I Program db2icrt completed successfully.

(0) root@Baltic # ./db2ilist
db2inst2
(0) root@Baltic # su - db2inst2
(0) db2inst2@Baltic $ db2start
04/21/2008 19:07:01    0    0    SQL1063N  DB2START processing was
successful.
SQL1063N  DB2START processing was successful.
```

---

*Example 12-61 Configuration of TCP/IP communication*

---

```
(0) db2inst2@Baltic $ db2set db2comm=tcPIP
(0) db2inst2@Baltic $ db2 update dbm cfg using svcename
DB2c_db2inst2
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
(0) db2inst2@Baltic $ cat /etc/services |grep DB2c_db2inst2
DB2c_db2inst2    50001/tcp
(0) db2inst2@Baltic $ db2stop
04/21/2008 19:07:00    0    0    SQL1064N  DB2STOP processing was
successful.
SQL1064N  DB2STOP processing was successful.
(0) db2inst2@Baltic $ db2start
04/21/2008 19:07:01    0    0    SQL1063N  DB2START processing was
successful.
SQL1063N  DB2START processing was successful.
```

---

**Note:** If you want to configure a shared disk HA on an existing DB2 instance, you have to move DB2 objects to the shared volume group. DB2 objects include instance home directory, diagpath, audit path as well as all the database objects.

3. Edit the `/etc/services` file.

The `SVCENAME` database manager configuration parameter defines the service name of the service port number that is listened by the DB2 instance. The `/etc/services` file on both nodes should have the same entries of DB2 listening ports. The `db2icrt` command adds the entries to the `/etc/services`



file on the owner node automatically. You have to add the port entries to the failover node manually. Example 12-62 shows the DB2 port entries in our lab systems.

*Example 12-62 The entries of /etc/services file on owner and failover nodes*

---

```
(O) root@node1 # cat /etc/services |grep db2inst2
DB2_db2inst2      61001/tcp
DB2_db2inst2_1   61002/tcp
DB2_db2inst2_2   61003/tcp
DB2_db2inst2_END 61004/tcp
(F) root@node2 # cat /etc/services |grep db2inst2
DB2_db2inst2      61001/tcp
DB2_db2inst2_1   61002/tcp
DB2_db2inst2_2   61003/tcp
DB2_db2inst2_END 61004/tcp
```

---

4. Confirm that the DB2 instance can move to another node manually.

This is a recommended sanity check. If the DB2 instances were not configured properly, the cluster domain might have unexpected behavior after including the DB2 instances.

Example 12-63 shows the process to shut down the owner node.

*Example 12-63 Shutdown process on owner node "Baltic"*

---

```
(O) db2inst2@Baltic $ db2_ps
Node 0
  UID      PID      PPID    C    STIME   TTY    TIME CMD
db2inst2  454896   114836  0    16:57:50 -      0:00 db2sysc 0
  root     69634    454896  0    16:57:51 -      0:00 db2ckpwd 0
  root     262382   454896  0    16:57:51 -      0:00 db2ckpwd 0
  root     368640   454896  0    16:57:51 -      0:00 db2ckpwd 0
(O) db2inst2@Baltic $ db2stop
04/22/2008 16:58:05 0 0 SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
(O) db2inst2@Baltic $ db2_ps
Node 0
  UID      PID      PPID    C    STIME   TTY    TIME CMD
db2inst2@Baltic $ exit
(O) root@Baltic # df |grep db2inst2
/dev/lvdb2inst2home 2621440 2254700 14% 327 1%
/home/db2inst2/db2home
/dev/lvdb2inst2db2 2621440 2384856 10% 76 1% /home/db2inst2/db2
(O) root@Baltic # umount /home/db2inst2/db2home
(O) root@Baltic # umount /home/db2inst2/db2
(O) root@Baltic # df |grep db2inst2
(O) root@Baltic # lspv |grep vg3
hdisk3          0009cdaa1628f8eb          vg3          active
```

```
(0) root@Baltic # varyoffvg vg3
(0) root@Baltic # lspv |grep vg3
hdisk3          0009cdaa1628f8eb          vg3
```

---

Example 12-64 shows the process to bring up DB2 on the failover node.

*Example 12-64 Start up process on failover node “Zaire”*

---

```
(F) root@Zaire # lspv |grep vg3
hdisk3          0009cdaa1628f8eb          vg3
(F) root@Zaire # varyonvg vg3
(F) root@Zaire # lspv |grep vg3
hdisk3          0009cdaa1628f8eb          vg3          active
(F) root@Zaire # mount /home/db2inst2/db2home
(F) root@Zaire # mount /home/db2inst2/db2
(F) root@Zaire # su - db2inst2
(F) db2inst2@Zaire $ echo "0 "$(hostname)" 0 "$(hostname)
0 Zaire 0 Zaire
(F) db2inst2@Zaire $ echo "0 "$(hostname)" 0 "$(hostname)
>~/sqlllib/db2nodes.cfg
(F) db2inst2@Zaire $ db2start
04/22/2008 17:05:29    0    0    SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

---

## Preparation for starting clustering

Two tasks on nodes are required before starting node clustering:

1. Preparing nodes for using db2haicu:

Before using the db2haicu utility, run the **preprnode** command on every node that is a part of the cluster domain. The command syntax is as follows:

```
(0) (F) # preprnode <node name> <node name>...
```

2. In our environment, as a root user, we issue the command shown in Example 12-65 on both nodes, Zaire and Baltic. You do not have to run this command for every DB2 instance, only once per node. Ensure that this command completes without any error.

*Example 12-65 Sample command of preprnode command*

---

```
(0) root@Baltic # preprnode Zaire Baltic
(F) root@Zaire # preprnode Zaire Baltic
```

---

## System clock synchronization

Synchronizing the date and time among nodes on cluster is not required, but we recommend doing this because synchronized clocks can make your problem determination more straightforward. When analyzing the logs on different nodes, you are able to see the time sequence of the events without any adjustment. Note that you can use the network time protocol (NTP) for this purpose. Refer to your operating system documentation for more information on how to configure NTP for your system.

## Creating an XML configuration file for db2haicu

Example 12-66 shows a sample XML file for a DB2 shared storage HA configuration. Refer to Figure 12-7 on page 538 and Table 12-2 on page 539 for the environment specific parameters. This XML file contains all the information that db2haicu needs:

- ▶ The <ClusterDomain> element:
  - This element covers all cluster-wide information, including quorum, cluster node, and cluster domain name.
  - The <PhysicalNetwork> sub-element
    - This element has all network related information including the network name and the network interface.
    - We have to define all network interfaces in this element whether we use the definition or not. We define a public network on network interface en1 and a private network on en1.
- ▶ The <FailoverPolicy> element:
  - This element specifies the failover type of the cluster nodes. Mutual means Active/Passive policy.
- ▶ The <DB2PartitionSet> element:
  - This element covers the DB2 instance information including the current DB2 instance name, the DB2 partition number, and the virtual IP address associated with the instance.
  - The <MutualPair> is the node pair which has been joined to the cluster domain. The attribute “systemPairNode1” is the active (owner) node, and the attribute “systemPairNode2” is the passive (failover) node.
- ▶ The <HADBSet> element:
  - This element specifies the database name to be configured to be highly available. It includes the current DB2 instance name.

*Example 12-66 XML configuration file for db2haicu*

---

```
<DB2Cluster
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="db2ha
a.xsd" clus
terManagerName="TSA" version="1.0">

  <ClusterDomain domainName="shared_disk_domain">
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="192.168.10.51"/>

    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
      <Interface interfaceName="en1" clusterNodeName="Baltic">
        <IPAddress baseAddress="192.168.10.48" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
      </Interface>
      <Interface interfaceName="en1" clusterNodeName="Zaire">
        <IPAddress baseAddress="192.168.10.45" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
      </Interface>
    </PhysicalNetwork>
    <PhysicalNetwork physicalNetworkName="db2_private_network_0"
physicalNetworkProtocol="ip">
      <Interface interfaceName="en0" clusterNodeName="Baltic">
        <IPAddress baseAddress="9.43.86.48" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
      </Interface>
      <Interface interfaceName="en0" clusterNodeName="Zaire">
        <IPAddress baseAddress="9.43.86.45" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
      </Interface>
    </PhysicalNetwork>
    <PhysicalNetwork physicalNetworkName="db2_private_network_1"
physicalNetworkProtocol="ip">
      <Interface interfaceName="en2" clusterNodeName="Baltic">
        <IPAddress baseAddress="192.168.20.48" subnetMask="255.255.252.0"
networkName="db2_private_network_1"/>
      </Interface>
      <Interface interfaceName="en2" clusterNodeName="Zaire">
        <IPAddress baseAddress="192.168.20.45" subnetMask="255.255.252.0"
networkName="db2_private_network_1"/>
      </Interface>
    </PhysicalNetwork>

    <ClusterNode clusterNodeName="Baltic"/>
    <ClusterNode clusterNodeName="Zaire"/>
  </ClusterDomain>

  <FailoverPolicy>
    <Mutual></Mutual>
  </FailoverPolicy>

  <DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst2">
```

```

        <VirtualIPAddress baseAddress="192.168.10.100" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
        <MutualPair systemPairNode1="Zaire" systemPairNode2="Baltic" />
    </DB2Partition>
</DB2PartitionSet>

    <HADBSet instanceName="db2inst2">
        <HADB databaseName="SAMPLE" />
    </HADBSet>

</DB2Cluster>

```

---

## Setting up a cluster domain using db2haicu with XML file

Before executing db2haicu, make sure that the DB2 instance has already started. db2haicu needs a running DB2 instance.

**Note:** We highly recommend that you execute db2haicu on the owner node. The *owner node* here is the node that is specified in the systemPairNode1 attribute. If you execute db2haicu on the failover node (the node specified in systemPairNode2 attribute), TSA tries to take over the DB2 resource group to the owner node as soon as db2haicu execution is complete.

Run db2haicu as DB2 instance owner. Here is the **db2haicu** command syntax:

```
(0) $ db2haicu -f <XML file name>
```

Example 12-67 shows a successful completion of the command execution.

*Example 12-67 Sample output of db2haicu execution using XML setup up mode*

---

```
(0) db2inst2@Zaire $ db2haicu -f shared_disk_domain.xml
```

```
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

```
db2haicu determined the current DB2 database manager instance is db2inst2. The cluster
configuration that follows will apply to this instance.
```

```
db2haicu is collecting information on your current setup. This step may take some time
as db2haicu will need to activate all databases for the instance to discover all paths
...
```

```
Creating domain shared_disk_domain in the cluster ...
Creating domain shared_disk_domain in the cluster was successful.
Configuring quorum device for domain shared_disk_domain ...
Configuring quorum device for domain shared_disk_domain was successful.
Adding network interface card en1 on cluster node Baltic to the network
db2_public_network_0 ...
Adding network interface card en1 on cluster node Baltic to the network
db2_public_network_0 was successful.
Adding network interface card en1 on cluster node Zaire to the network
db2_public_network_0 ...
Adding network interface card en1 on cluster node Zaire to the network
db2_public_network_0 was successful.
Adding network interface card en0 on cluster node Baltic to the network
db2_private_network_0 ...
Adding network interface card en0 on cluster node Baltic to the network
db2_private_network_0 was successful.
Adding network interface card en0 on cluster node Zaire to the network
db2_private_network_0 ...
Adding network interface card en0 on cluster node Zaire to the network
db2_private_network_0 was successful.
Adding network interface card en2 on cluster node Baltic to the network
db2_private_network_1 ...
Adding network interface card en2 on cluster node Baltic to the network
db2_private_network_1 was successful.
Adding network interface card en2 on cluster node Zaire to the network
db2_private_network_1 ...
Adding network interface card en2 on cluster node Zaire to the network
db2_private_network_1 was successful.
Adding DB2 database partition 0 to the cluster ...
Adding DB2 database partition 0 to the cluster was successful.
Adding database SAMPLE to the cluster domain ...
Adding database SAMPLE to the cluster domain was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
```

---

## Verifying the created cluster domain

The TSA command **issam** and the DB2 command **db2pd** are useful for verifying DB2 clustering with TSA. **issam** lists all the resource groups and resources contained in the resource groups as well as the information about operational states of these objects. The command syntax is as follows:

```
(0) (F) # issam
```

Refer to 8.4.2, “Configuration of TSA and DB2” on page 246 for more information about the **issam** command.

Example 12-68 shows the resource group, `db2_db2inst2_0-rg`, is online and the active node is Zaire that owns all resources in this cluster domain.

*Example 12-68 Sample output of lssam command*

**(0) (F) \$ lssam**

```
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_0-rs
   |- Offline IBM.Application:db2_db2inst2_0-rs:Baltic
   '- Online IBM.Application:db2_db2inst2_0-rs:Zaire
|- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
   |- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
   '- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
|- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
   |- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
   '- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
'- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
   |- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
   '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

In DB2 9.5, the **db2pd** command option **-ha** provides data of the cluster domain:

(0) \$ db2pd -ha

Example 12-69 shows the output of this command.

*Example 12-69 Sample output for db2pd -ha command*

**(0) \$ db2pd -ha**

```
DB2 HA Status
Instance Information:
Instance Name           = db2inst2
Number Of Domains      = 1
Number Of RGs for instance = 1

Domain Information:
Domain Name             = shared_disk_domain
Cluster Version         = 2.4.8.3
Cluster State           = Online
Number of nodes         = 2

Node Information:
Node Name              State
-----
Baltic                 Online
Zaire                  Online

Resource Group Information:
Resource Group Name    = db2_db2inst2_0-rg
Resource Group LockState = Unlocked
Resource Group OpState = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 4
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  Baltic
  Zaire

Member Resource Information:
Resource Name          = db2mnt-home_db2inst2_db2-rs
Resource State         = Online
Resource Type          = Mount
```

```

Mount Resource Path          = /home/db2inst2/db2
Number of Allowed Nodes     = 2
  Allowed Nodes
  -----
  Baltic
  Zaire

Resource Name                = db2ip_192_168_10_121-rs
Resource State               = Online
Resource Type                = IP

Resource Name                = db2_db2inst2_0-rs
Resource State               = Online
Resource Type                = DB2 Partition
DB2 Partition Number        = 0
Number of Allowed Nodes     = 2
  Allowed Nodes
  -----
  Baltic
  Zaire

Resource Name                = db2mnt-home_db2inst2_db2home-rs
Resource State               = Online
Resource Type                = Mount
Mount Resource Path         = /home/db2inst2/db2home
Number of Allowed Nodes     = 2
  Allowed Nodes
  -----
  Baltic
  Zaire

Network Information:
Network Name                  Number of Adapters
-----
db2_private_network_1        2

  Node Name                    Adapter Name
  -----
  Baltic                        en2
  Zaire                          en2

Network Name                  Number of Adapters
-----
db2_private_network_0        2

  Node Name                    Adapter Name
  -----
  Baltic                        en0
  Zaire                          en0

Network Name                  Number of Adapters
-----
db2_public_network_0         2

  Node Name                    Adapter Name
  -----
  Baltic                        en1
  Zaire                          en1

Quorum Information:
Quorum Name                  Quorum State
-----
Fail                          Offline
db2_Quorum_Network_192_168_10_51:13_35_7  Online
Operator                      Offline

```

---



## 12.4.3 Administration

From time to time, you need to perform some maintenance tasks on the node in the shard disk HA cluster domain, such as upgrading software. In this section we demonstrate how to detach one node from the cluster domain and how to stop the cluster domain for system maintenance.

### The node maintenance scenario

You can utilize the idle state of the failover node to perform the maintenance tasks on that node without stopping the entire cluster domain. The process is detaching the failover node from the cluster, perform the maintenance work, then re-attach the node back to the cluster. If the maintenance work is to be performed on the owner node, you need to swap the role of the node first.

#### 1. Swap the owner node:

If the node to be maintained is the owner node, switch its role to be the failover. Use the **rgreq** command to move the resource groups on the owner node.

```
(0) # rgreq -o move <resource group>
```

Verify that all resource groups are moved to the new owner node using the **lssam** command.

#### 2. Detach the failover node:

Use the **samctrl** command to detach the maintenance node as follows:

```
(0) # samctrl -u a <node to detach>
```

Example 12-70 shows the execution of the **samctrl** command and the system status after the operation.

*Example 12-70 Sample of the samctrl command and the output*

---

```
(0) root@Zaire # samctrl -u a Baltic
(0) root@Zaire # lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_0-rs
|  |- Offline IBM.Application:db2_db2inst2_0-rs:Baltic Node=Excluded
|  ' - Online IBM.Application:db2_db2inst2_0-rs:Zaire
|- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
|  |- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic Node=Excluded
|  ' - Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
|- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
|  |- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic Node=Excluded
|  ' - Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
' - Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
|  |- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic Node=Excluded
|  ' - Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

### 3. Perform maintenance tasks:

You can perform the maintenance tasks on the failover node after it is detached from the cluster domain. For example, you can reboot OS or upgrade software.

**Note:** When you shut down the RSCT or TSA on the detached node, the output of the `lsam` command shows *Failed Offline* for the node. There is no impact to the cluster node. But remember that the output is changed.

### 4. Reintegration to the cluster domain:

When you would like to reintegrate the detached node, you should execute the `samctrl` command again:

```
(O) # samctrl -u d <node to detach>
```

After you execute the `samctrl` command, ensure that there is no *Excluded* mode in the output of the `lsam` command.

## Cluster maintenance

When you need to work on DB2 instances or all the resources, you have to stop the cluster domain. For example, you have to stop the cluster domain when you migrate the DB2 to next version. Use the `chrg` command and `stoprpdomain` command to stop the cluster domain.

Perform these steps to stop the cluster domain:

#### 1. Change the nominal state of the resource groups to offline.

Changing the nominal status of the resource group to *offline* causes TSA to try to stop the all resources in the resource group. After that, you can stop the cluster domain:

```
(O) # chrg -o offline <resource group>
```

After you execute the `chrg` command, ensure that all resource group and resources in this domain are offline by using the `lsam` command.

#### 2. Stop the cluster domain.

Use `stoprpdomain` to stop the cluster domain.

```
(O) # stoprpdomain <domain>
```

Example 12-71 shows that the cluster domain `shared_disk_domain` is stopped.

*Example 12-71 Sample command of stopping the cluster domain*

---

```
(0) # root@Zaire # stoprpdomain shared_disk_domain
(0) # root@Zaire # lsrpdomain
Name                OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
shared_disk_domain  Offline  2.4.8.3            No              12347   12348
```

---

3. Execute maintenance.

You can perform the maintenance task on this nodes, for example, to migrate the DB2 or TSA.

4. Start the cluster domain.

After the maintenance, re-start the cluster domain using **starttrpdomain**:

```
(0) # starttrpdomain <domain>
```

Example 12-72 shows the that the shared\_disk\_domain is back online.

*Example 12-72 Sample command of stopping the cluster domain*

---

```
root@Zaire # starttrpdomain shared_disk_domain
root@Zaire # lsrpdomain
Name                OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
shared_disk_domain  Online   2.4.8.3            No              12347   12348
```

---

5. Change the nominal state of the resource groups to online.

Use the **chrg** command to change the nominal state of the resource groups:

```
(0) # chrg -o online <resource group>
```

After you execute the **chrg** command, verify that the resource groups and all resources in this domain are online.

## 12.4.4 Testing

In this section we show some resource failure tests to verify the HA cluster setup:

- ▶ Operating system failure
- ▶ Power failure
- ▶ Network failure
- ▶ DB2 instance failure

We suggest that you perform similar tests before deploying the HA configuration to production.

## Operating system failure

We use the operating system restart command **shutdown -Fr** to simulate the operating system failure situation. When the owner node restart happens, the failover node takes over the resource group and all resources. On the other hand, if the failover node fails, the resource group and all resources remain in the owner node. After the failover node is recovered, the cluster domain reintegrates the failover node again automatically.

The next four examples show the transition of the cluster domain in the case of the owner node failure.

Example 12-73 shows the state of the cluster domain before shutdown starts. Baltic is online and owns the resources.

### *Example 12-73 Cluster domain state before shutdown starts*

---

```
$ lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_0-rs
    |- Online IBM.Application:db2_db2inst2_0-rs:Baltic
    '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

The **shutdown -Fr** command is issued on the owner node Baltic to simulate the operating system failure. Example 12-74 shows that the failover is taking place at the cluster domain starting failover after shutdown starts.

### *Example 12-74 Failover is taking place*

---

```
Pending online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Offline IBM.Application:db2_db2inst2_0-rs
    |- Failed offline IBM.Application:db2_db2inst2_0-rs:Baltic Node=Offline
    '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Failed offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic Node=Offline
    '- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Pending online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Failed offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic Node=Offline
    '- Pending online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Pending online IBM.Application:db2mnt-home_db2inst2_db2home-rs
    |- Failed offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic Node=Offline
    '- Pending online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

Example 12-75 shows that TSA has failed over the resources to the failover node Zaire.

*Example 12-75 Failover completed*

---

```
Sat Apr 19 22:02:51 PDT 2008
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_0-rs
|   |- Failed offline IBM.Application:db2_db2inst2_0-rs:Baltic Node=Offline
|   '- Online IBM.Application:db2_db2inst2_0-rs:Zaire
|- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
|   |- Failed offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic Node=Offline
|   '- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
|- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
|   |- Failed offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic Node=Offline
|   '- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
'- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
|   |- Failed offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic Node=Offline
|   '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

Example 12-76 shows that the cluster domain completes re-integrating the old owner node.

*Example 12-76 Old owner node re-integrated*

---

```
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst2_0-rs
|   |- Offline IBM.Application:db2_db2inst2_0-rs:Baltic
|   '- Online IBM.Application:db2_db2inst2_0-rs:Zaire
|- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
|   |- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
|   '- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
|- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
|   |- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
|   '- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
'- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
|   |- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
|   '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

## Power failure

The power failure can be simulated by simply unplugging the power cable. The failover process is very similar to the operating system failure. **Issam** output is similar as well. This test can reassure you that the failover behavior of the cluster domain is as expected.

## Network failure

Network failure test checks the HA configuration of the service IP address. We simulate the network failure by unplugging the network cable of the public network in the owner node. In our environment it is the cable of the en1 network interface in Baltic.

The next three examples show the transition of the cluster domain on network failure.

Example 12-77 shows the cluster domain state before the network failure test starts.

*Example 12-77 Cluster domain before network failure test starts*

---

```
# lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_0-rs
    |- Online IBM.Application:db2_db2inst2_0-rs:Baltic
    '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

We unplug the network cable. Example 12-78 shows the cluster domain starting to fail over the resource from the owner node to the failover node.

*Example 12-78 Cluster domain starts failover process*

---

```
# lssam
Pending offline IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Pending offline IBM.Application:db2_db2inst2_0-rs
    |- Pending offline IBM.Application:db2_db2inst2_0-rs:Baltic
    '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
    '- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire
```

---

Example 12-79 shows that the cluster domain completes the failover process. Zaire is now the owner node.

*Example 12-79 Failover completed*

---

```
# lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_0-rs
```

```

        |- Offline IBM.Application:db2_db2inst2_0-rs:Baltic
        '- Online IBM.Application:db2_db2inst2_0-rs:Zaire
|- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
    '- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
|- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
    '- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
'- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
    |- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
    '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire

```

---

## DB2 instance failure

This test checks the DB2 instance failure by stopping the DB2 process. When TSA detects the failure of the resources, TSA tries to start the resource on the same node once. So we expect that this failure is recovered on the same node in this case.

When db2haicu creates the cluster domain, it registers the scripts for starting, stopping, and monitoring resources. TSA detects the failure by the return code (RC) of the monitor script executed by TAS. TSA takes actions based on the RC. If the start script completes successfully with RC=0 and the next return code from the monitor script is RC=1 (online), TSA determines that the node has recovered by itself successfully. If the start script returns a non-0 return code (RC<>0) or the next monitor script return code is RC=2 (offline), TSA starts the standby node takeover process immediately. In this test case, the monitor script returns RC=2 (offline), which triggers the TSA to start the failed resource (DB2 processes).

To simulate a DB2 instance failure, we stop the DB2 process by using the `kill` command:

```
(0) # kill <process ID of DB2>
```

The next three examples show the recovery progress of the Baltic. Example 12-80 shows the cluster domain state before the DB2 process failed.

*Example 12-80 Initial cluster domain state*

---

```

# lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_0-rs
    |- Online IBM.Application:db2_db2inst2_0-rs:Baltic
    '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
    '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic

```

```

        '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
    '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
        |- Online
IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
        '- Offline
IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire

```

---

We stop a DB2 process as follows:

```

# ps -ef|grep db2sysc |grep db2inst2
db2inst2 733224 123060  0 14:32:41      - 0:16 db2sysc 0
# kill -9 733224

```

Example 12-81 shows that the recovery is started on the same node (Baltic).

*Example 12-81 The Recovery is started on Baltic*

```

# lssam
Pending online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst2_0-rs
        |- Pending online IBM.Application:db2_db2inst2_0-rs:Baltic
        '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
        |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
        '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
        |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
        '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
        |- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
        '- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire

```

---

Example 12-82 shows that the restart is successful on Baltic.

*Example 12-82 Self node restart successful*

```

# lssam
Online IBM.ResourceGroup:db2_db2inst2_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst2_0-rs
        |- Online IBM.Application:db2_db2inst2_0-rs:Baltic
        '- Offline IBM.Application:db2_db2inst2_0-rs:Zaire
  |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs
        |- Online IBM.ServiceIP:db2ip_192_168_10_121-rs:Baltic
        '- Offline IBM.ServiceIP:db2ip_192_168_10_121-rs:Zaire
  |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs
        |- Online IBM.Application:db2mnt-home_db2inst2_db2-rs:Baltic
        '- Offline IBM.Application:db2mnt-home_db2inst2_db2-rs:Zaire
  '- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs
        |- Online IBM.Application:db2mnt-home_db2inst2_db2home-rs:Baltic
        '- Offline IBM.Application:db2mnt-home_db2inst2_db2home-rs:Zaire

```

---



## 12.5 DB2 single partition HA with shared storage on Linux

In this section we present a test case that uses the new DB2 9.5 HA feature to automate failover of a database. We describe the HA configuration for a single partition database on shared storage in this section. For an HA configuration in a DPF environment, refer to 12.6, “DB2 DPF HA configuration with shared disk on AIX” on page 580.

### 12.5.1 Architecture

The example shows an active/passive hot standby configuration, and assumes that the following minimum configuration is already in place:

- ▶ Two servers are running SLES 10 SP1 x64.
- ▶ There is one network interface card per server.
- ▶ There is a common network access to a third gateway device as a tiebreaker.
- ▶ DB2 9.5 FixPack1 with SA MP Base Component is installed.
- ▶ An IBM System Storage DS4700 storage server is connected to two Linux servers and all LUNs are allocated already. For more information about how to configure and use the DS4700 storage server, refer to *IBM System Storage DS4000 Series and Storage Manager*, SG24-7010.
- ▶ DB2 9.5 client is running on Windows XP.

Table 12-3 shows the resources and naming convention used in this scenario.

Table 12-3 Resources and name convention

Server	mensa	lepus
IP address	9.43.86.90	9.43.86.91
Local instance	db2inst1	
Instance home file system	/home/db2inst1/db2home	
Database data file system	/home/db2inst1/db2data	
Database log file system	/home/db2inst1/db2log	
Diagnostic log file system	/home/db2inst1/db2dump	
DB2 audit log file system	/home/db2inst1/db2audit	
Service IP address	9.43.86.252	
Database name	TEST	
HA role	Active	Passive

Figure 12-9 shows the architecture.

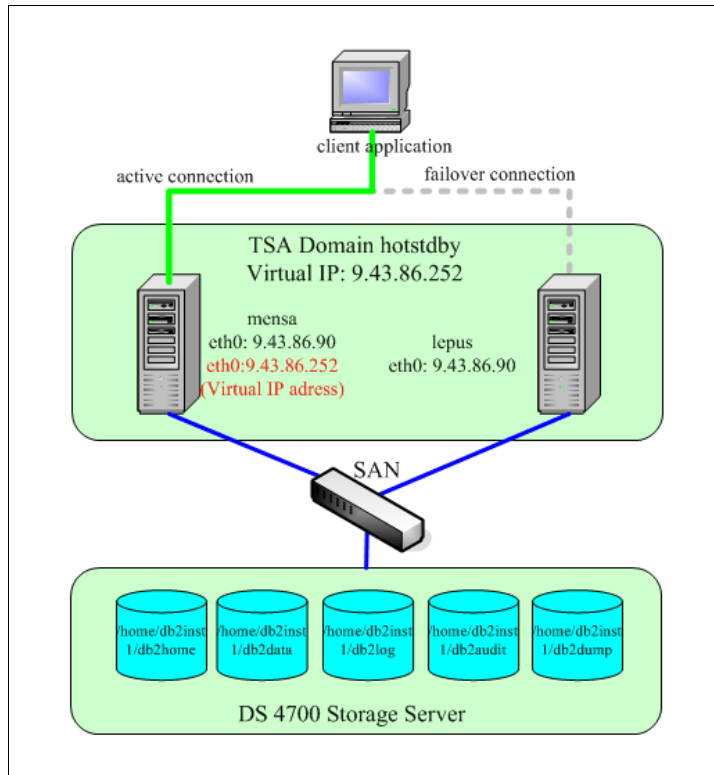


Figure 12-9 Single network shared storage architecture

**Note:** In this scenario, each server has only one network adapter connected through one switch. We highly recommend that you use two network adapters and switches to improve network resilience and availability.

## 12.5.2 Configuration

In this section, we show a step-by-step procedure to create a DB2 instance, create a database, and make them highly available through the db2haicu utility.

## Setting up the network

Each of the two servers contains one network interface(eth0). To set up the required connections between the servers, follow these steps:

1. Assign IP addresses to active and passive nodes.

The eth0 network interfaces are connected to each other through the external network cloud forming the public network. We assigned the following static IP addresses to the eth0 adapter on the active and passive nodes:

```
Active node mensa: 9.43.86.90
Passive node lepus: 9.43.86.91
```

2. Map host names.

Make sure that the host names are mapped to their corresponding public IP addresses in the /etc/hosts file:

```
9.43.86.90      mensa.itsosj.sanjose.ibm.com mensa
9.43.86.91      lepus.itsosj.sanjose.ibm.com lepus
```

Defining the host names in the /etc/hosts file allows the nodes to identify each other through host name. All cluster nodes should have the same entries in the /etc/hosts file and have fixed IP addresses.

Check /etc/nsswitch file, make sure files is the first one used for hosts name services. In this example, the hosts entries in /etc/nsswitch file is as follows:

```
hosts:          files dns
```

3. The active and passive nodes should be able to ping each other on public network. Issue the **ping -c 1 *hostname*** commands on both nodes and make sure that they complete successfully. See Example 12-83.

*Example 12-83 Make sure host name resolution works*

---

```
mensa:/ # ping -c 1 lepus
PING lepus.itsosj.sanjose.ibm.com (9.43.86.91) 56(84) bytes of data.
64 bytes from lepus.itsosj.sanjose.ibm.com (9.43.86.91): icmp_seq=1 ttl=64 time=0.122
ms

--- lepus.itsosj.sanjose.ibm.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.122/0.122/0.122/0.000 ms
mensa:/ # ping -c 1 mensa
PING mensa.itsosj.sanjose.ibm.com (9.43.86.90) 56(84) bytes of data.
64 bytes from mensa.itsosj.sanjose.ibm.com (9.43.86.90): icmp_seq=1 ttl=64 time=0.019
ms

--- mensa.itsosj.sanjose.ibm.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.019/0.019/0.019/0.000 ms
lepus:~ # ping -c 1 mensa
PING mensa.itsosj.sanjose.ibm.com (9.43.86.90) 56(84) bytes of data.
```

```
64 bytes from mensa.itsosj.sanjose.ibm.com (9.43.86.90): icmp_seq=1 ttl=64 time=0.131 ms
```

```
--- mensa.itsosj.sanjose.ibm.com ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
rtt min/avg/max/mdev = 0.131/0.131/0.131/0.000 ms
```

```
lepus:~ # ping -c 1 mensa
```

```
PING mensa.itsosj.sanjose.ibm.com (9.43.86.90) 56(84) bytes of data.
```

```
64 bytes from mensa.itsosj.sanjose.ibm.com (9.43.86.90): icmp_seq=1 ttl=64 time=0.124 ms
```

```
--- mensa.itsosj.sanjose.ibm.com ping statistics ---
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
rtt min/avg/max/mdev = 0.124/0.124/0.124/0.000 ms
```

---

## Configuring shared file systems

In SLES 10, you can use the traditional partitioning tool to manage disk devices, or you can use multipath I/O to manage them in the Fibre Channel (FC) or iSCSI SAN environments. In this example, we use multipath I/O and LVM2 to manage the shared storage. For more information about SLES 10 multipath I/O, refer to:

[http://www.novell.com/documentation/sles10/stor\\_evms/data/multipathing.html](http://www.novell.com/documentation/sles10/stor_evms/data/multipathing.html)

**Note:** In SLES 10, you can also use Enterprise Volume Management System (EVMS) to manage storage. For more information about EVMS, refer to: [http://evms.sourceforge.net/users\\_guide/#intro](http://evms.sourceforge.net/users_guide/#intro).

### *Configure shared file systems on the active node*

After you finish the multipathing configuration, check the `/dev/disk/by-name` directory. You can see some multipathing devices, Example 12-84 is an output in our environment.

*Example 12-84 Sample output of /dev/disk/by-name*

---

```
mena:/dev/disk/by-name # ll
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath1 -> ../../dm-0
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath10 -> ../../dm-9
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath2 -> ../../dm-1
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath3 -> ../../dm-2
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath4 -> ../../dm-3
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath5 -> ../../dm-4
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath6 -> ../../dm-5
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath7 -> ../../dm-6
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath8 -> ../../dm-7
lrwxrwxrwx 1 root root 10 Apr 10 22:11 mpath9 -> ../../dm-8
```

---

You can check information about multipathed devices through the `multipath -ll` command. Example 12-85 shows a sample output in our environment.

*Example 12-85 Information of multipathed devices*

---

```

mensa:/dev/disk/by-name # multipath -ll
mpath2 (360050768018600c4700000000000091) dm-1 IBM,2145
[size=150G][features=0][hwhandler=0]
\_ round-robin 0 [prio=100][active]
  \_ 4:0:0:0 sdc 8:32 [active][ready]
  \_ 4:0:2:0 sde 8:64 [active][ready]
\_ round-robin 0 [prio=20][enabled]
  \_ 4:0:1:0 sdd 8:48 [active][ready]
  \_ 4:0:3:0 sdf 8:80 [active][ready]
mpath1 (SATA_WDC_WD5000AAKS-_WD-WCAPW0990173) dm-0 ATA,WDC WD5000AAKS-6
[size=466G][features=0][hwhandler=0]
\_ round-robin 0 [prio=0][active]
  \_ 2:0:0:0 sdb 8:16 [active][ready]
mpath9 (3600a0b800026b2820000307f47f4d5a7) dm-8 IBM,VirtualDisk
[size=29G][features=0][hwhandler=0]
\_ round-robin 0 [prio=0][active]
  \_ 5:0:0:6 sdm 8:192 [active][ready]
mpath8 (3600a0b80002904de0000342247f4d65a) dm-7 IBM,VirtualDisk
[size=29G][features=0][hwhandler=0]
\_ round-robin 0 [prio=0][active]
  \_ 5:0:0:5 sdl 8:176 [active][ready]
...

```

---

Now we create volume groups (VGs), logical volumes (LVs), and file systems with multipathed devices. You can do this with the YaST GUI or from the command line. In this example, we use the ext3 file system. Table 12-4 lists the storage resources we created for this scenario.

*Table 12-4 Storage resources*

Multipathed devices	Volume groups	Logical volumes	File systems
/dev/mapper/mpath3	db2homevg	db2homeLv	/home/db2inst1/db2home
/dev/mapper/mpath4	db2datavg	db2dataLv1	/home/db2inst1/db2data
		db2logLv1	/home/db2inst1/db2log
		db2audit	/home/db2inst1/db2audit
		db2dump	/home/db2inst1/db2dump

**Note:** In this example, we only use one datavg volume group for data file systems and log file systems. However, we recommend that you use separate volume groups for data and log file systems. It is beneficial to separate them into different FCs too. These measures can improve database performance and availability.

Mount the file systems manually through the **mount** command. Example 12-86 shows the entries in the `/etc/fstab` file. Note that you should use the **noauto** mount option, disable the file systems dump, and **fsck** by specifying 0 for the fifth and sixth fields.

*Example 12-86 Mount points in /etc/fstab*

---

<code>/dev/datavg/db2data1v1</code>	<code>/home/db2inst1/db2data</code>	<code>ext3</code>	<code>acl,user_xattr,noauto</code>	<code>0</code>	<code>0</code>
<code>/dev/datavg/db2log1v1</code>	<code>/home/db2inst1/db2log</code>	<code>ext3</code>	<code>acl,user_xattr,noauto</code>	<code>0</code>	<code>0</code>
<code>/dev/db2homevg/db2home1v</code>	<code>/home/db2inst1/db2home</code>	<code>ext3</code>	<code>acl,user_xattr,noauto</code>	<code>0</code>	<code>0</code>
<code>/dev/datavg/db2audit</code>	<code>/home/db2inst1/db2audit</code>	<code>ext3</code>	<code>acl,user_xattr,noauto</code>	<code>0</code>	<code>0</code>
<code>/dev/datavg/db2dump</code>	<code>/home/db2inst1/db2dump</code>	<code>ext3</code>	<code>acl,user_xattr,noauto</code>	<code>0</code>	<code>0</code>

---

**Configure shared file systems on the passive node**

Finish the multipathing configuration for Linux as you do on the active node. Because we create all VGs, LVs, and file systems on the active node, we only need to import these definitions on the passive node and add proper mount points.

Follow these steps to configure shared file systems on the passive node:

1. Export VGs from the active node. Example 12-87 shows the commands to do this in our example.

*Example 12-87 Command to export VGs*

---

```
mensa:/ # umount /home/db2inst1/db2home
mensa:/ # umount /home/db2inst1/db2data
mensa:/ # umount /home/db2inst1/db2log
mensa:/ # umount /home/db2inst1/db2audit
mensa:/ # umount /home/db2inst1/db2dump
mensa:/ # vgchange -an db2datavg
mensa:/ # vgchange -an db2homevg
mensa:/ # vgexport db2datavg
mensa:/ # vgexport db2homevg
```

---

2. Import and activate VGs in passive node: Example 12-88 shows the command.

### *Example 12-88 Command to import and activate VGs*

---

```
lepus:/ # pvscan
lepus:/ # vgscan
lepus:/ # vgimport db2datavg
lepus:/ # vgimport db2homevg
lepus:/ # vgchange -ay db2datavg
lepus:/ # vgchange -ay db2homevg
```

---

3. Modify the `/etc/fstab` file to add mount points as shown in Example 12-86.
4. Check if we can mount these file systems correctly.
5. Unmount these file systems on the passive node.
6. Activate VGs and mount the file systems on the active node (Example 12-89).

### *Example 12-89 Activate VGs and mount file systems*

---

```
lepus:/ # vgchange -ay db2datavg
lepus:/ # vgchange -ay db2homevg
lepus:/ # mount /home/db2inst1/db2home
lepus:/ # mount /home/db2inst1/db2data
lepus:/ # mount /home/db2inst1/db2log
lepus:/ # mount /home/db2inst1/db2audit
lepus:/ # mount /home/db2inst1/db2dump
```

---

**Note:** Now all VGs are defined and activated in both nodes. Make sure that the file systems are mounted only in one server at one time.

## **Creating a DB2 instance on the active node**

These steps show how to create a DB2 instance on the active node mensa:

1. Before creating the DB2 instance, make sure that the file systems are mounted properly, as shown in Example 12-90 for all five file systems.

### *Example 12-90 Check file systems mount*

---

```
mena:/ # mount
/dev/sda1 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
nfsd on /proc/fs/nfsd type nfsd (rw)
/dev/mapper/datavg-db2data1v1 on /home/db2inst1/db2data type ext3 (rw,acl,user_xattr)
/dev/mapper/datavg-db2log1v1 on /home/db2inst1/db2log type ext3 (rw,acl,user_xattr)
/dev/mapper/db2homevg-db2home1v on /home/db2inst1/db2home type ext3 (rw,acl,user_xattr)
/dev/mapper/datavg-db2dump on /home/db2inst1/db2dump type ext3 (rw,acl,user_xattr)
/dev/mapper/datavg-db2audit on /home/db2inst1/db2audit type ext3 (rw,acl,user_xattr)
```

---

2. Create groups and users for the DB2 instance. Example 12-91 shows the commands.

*Example 12-91 Create groups and users*

---

```
mensa:/ # groupadd -g 107 db2iadm1
mensa:/ # groupadd -g 108 db2fadm1
mensa:/ # useradd -d /home/db2inst1/db2home -g 107 db2inst1
mensa:/ # useradd -d /home/db2fenc1 -m -g 108 db2fenc1
```

---

3. Change the ownership of the five systems that db2inst1 uses. Example 12-92 shows the commands:

*Example 12-92 Change ownership of file systems*

---

```
mensa:/ # chown db2inst1:db2iadm1 /home/db2inst1/db2home
mensa:/ # chown db2inst1:db2iadm1 /home/db2inst1/db2data
mensa:/ # chown db2inst1:db2iadm1 /home/db2inst1/db2log
mensa:/ # chown db2inst1:db2iadm1 /home/db2inst1/db2audit
mensa:/ # chown db2inst1:db2iadm1 /home/db2inst1/db2dump
```

---

4. Run the **db2icrt** command as root to create DB2 instance db2inst1 as shown in Example 12-93.

*Example 12-93 Create DB2 instance*

---

```
mensa:/ # cd /opt/ibm/db2/V9.5/instance
mensa:/opt/ibm/db2/V9.5/instance # ./db2icrt -u db2fenc1 db2inst1
DBI1070I Program db2icrt completed successfully.
mensa:/opt/ibm/db2/V9.5/instance # ./db2ilist
db2inst1
```

---

5. Make sure that the DB2 instance db2inst1 can be started successfully.
6. Unmount the file systems on the active node as shown in Example 12-87 on page 564. Do not deactivate and export VGs.

## Creating a DB2 instance on the passive node

Follow the same procedure shown in “Creating a DB2 instance on the active node” on page 565 to create DB2 instance on the passive node. In our scenario, it is lepus. Note that before creating DB2 instance, we delete the directory /home/db2inst1/db2home/sqllib, because we share the home directory with the active node. /home/db2inst1/db2home/sqllib is created when we create the DB2 instance on the active node. Otherwise, we encounter an error.

After successfully creating the DB2 instance db2inst1, we unmount the file systems on the passive node and remount the file systems on the active node. We are now ready to configure HA cluster through db2haicu.



**Note:** You can configure the HA cluster from either node. We configure it from the active node mensa.

## Setting up HA cluster through db2haicu utility

This section demonstrates a step-by-step procedure to configure the HA cluster through the db2haicu utility. All commands run on the active node mensa if no specific declaration is made.

### **HA cluster domain preparation**

Before using the db2haicu tool, both nodes must be prepared to have proper security. With root authority, run the command **preprnode** as shown in Example 12-94 on mensa and lepus.

*Example 12-94 Prepare node for SA MP cluster*

---

```
mensa:/ # /usr/sbin/rsct/bin/preprnode mensa lepus
...
lepus:/ # /usr/sbin/rsct/bin/preprnode lepus mensa
```

---

This command only needs to be run once per node but not for every DB2 instance that is to be made highly available.

### **Set up HA cluster through db2haicu interactive setup mode**

After completing the preceding preliminary configuration, we can use the db2haicu tool to automate HA configuration. The db2haicu must be run on the node hosting the DB2 instance, and the DB2 instance must have already started. The details are outlined in 12.1, “db2haicu” on page 478.

The steps are as follows:

1. Create a cluster domain.

Switch the user to db2inst1, start the DB2 instance, then issue the **db2haicu** command. The output should look as shown in Example 12-95.

*Example 12-95 Create a cluster domain using db2haicu*

---

```
db2inst1@mensa:~> db2start
04/11/2008 14:30:47    0    0    SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
db2inst1@mensa:~> db2haicu
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst1. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

When you use db2haicu to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with db2haicu' in the DB2 Information Center. db2haicu is searching the current machine for an existing active cluster domain ...

db2haicu did not find a cluster domain on this machine. db2haicu will now query the system for information about cluster nodes to create a new cluster domain ...

db2haicu did not find a cluster domain on this machine. To continue configuring your clustered environment for high availability, you must create a cluster domain; otherwise, db2haicu will exit.

Create a domain and continue? [1]

1. Yes
  2. No
- 

**Note:** The number contained within square brackets is the default for that prompt. Simply press Enter to select the default value.

In the “Create a domain and continue” prompt, press Enter to continue. It prompts for the cluster domain name. Input the name hotstdby; then it prompts for “How many cluster nodes will the domain hotstdby contain”. In this case, input 2, then input the host name mensa and lepus subsequently. After that, you can press Enter to create the domain. Example 12-96 shows these sequences from db2haicu.

*Example 12-96 Create a cluster domain using db2haicu*

---

Create a unique name for the new domain:

**hotstdby**

Nodes must now be added to the new domain.

How many cluster nodes will the domain hotstdby contain?

**2**

Enter the host name of a machine to add to the domain:

**mensa**

Enter the host name of a machine to add to the domain:

**lepus**

db2haicu can now create a new domain containing the 2 machines that you specified. If you choose not to create a domain now, db2haicu will exit.

Create the domain now? [1]

1. Yes
2. No

```
Creating domain hotstdby in the cluster ...  
Creating domain hotstdby in the cluster was successful.
```

---

## 2. Configure a quorum device.

After we create the domain, we must configure a quorum for the cluster domain. The supported quorum type for this solution is a network quorum. A network quorum (or network tiebreaker) is a pingable IP address that is used to decide which node in the cluster serves as the active node during a site failure, and which nodes are offline. Note that the machine hosting this IP address does not require any particular software nor operating system level installed; the primary requirement is that it is pingable from all nodes in the cluster, and remains pingable in the case of cluster node failures.

db2haicu prompts to enter the quorum configuration values as shown in Example 12-97. In this case, we use 9.43.85.1.

### *Example 12-97 Configure a network quorum device*

---

You can now configure a quorum device for the domain. For more information, see the topic "Quorum devices" in the DB2 Information Center. If you do not configure a quorum device for the domain, then a human operator will have to manually intervene if subsets of machines in the cluster lose connectivity.

```
Configure a quorum device for the domain called hotstdby? [1]  
1. Yes  
2. No
```

The following is a list of supported quorum device types:

```
1. Network Quorum  
Enter the number corresponding to the quorum device type to be used: [1]
```

Specify the network address of the quorum device:

```
9.43.85.1  
Configuring quorum device for domain hotstdby ...  
Configuring quorum device for domain hotstdby was successful.
```

---

## 3. Set up the network for the cluster domain.

After defining the quorum device, db2haicu prompts to create networks for the cluster domain. If network failure detection is important to your configuration, you must follow the prompts and add the networks to the cluster at this point. db2haicu discovers all network interfaces automatically.

In this example, we create one public network and add eth0 network interface to this public network. Example 12-98 shows the sample screen output.

### *Example 12-98 Create networks*

---

Create networks for these network interface cards? [1]

1. Yes
2. No

Enter the name of the network for the network interface card: eth0 on cluster node:  
mensa.itsosj.sanjose.ibm.com

1. Create a new public network for this network interface card.
2. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node  
mensa.itsosj.sanjose.ibm.com to the network db2\_public\_network\_0? [1]

1. Yes
2. No

Adding network interface card eth0 on cluster node mensa.itsosj.sanjose.ibm.com to  
the network db2\_public\_network\_0 ...

Adding network interface card eth0 on cluster node mensa.itsosj.sanjose.ibm.com to  
the network db2\_public\_network\_0 was successful.

Enter the name of the network for the network interface card: eth0 on cluster node:  
lepus.itsosj.sanjose.ibm.com

1. db2\_public\_network\_0
2. Create a new public network for this network interface card.
3. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node  
lepus.itsosj.sanjose.ibm.com to the network db2\_public\_network\_0? [1]

1. Yes
2. No

Adding network interface card eth0 on cluster node lepus.itsosj.sanjose.ibm.com to  
the network db2\_public\_network\_0 ...

Adding network interface card eth0 on cluster node lepus.itsosj.sanjose.ibm.com to  
the network db2\_public\_network\_0 was successful.

---

#### 4. Select cluster manager.

After the network definitions, db2haicu prompts for the Cluster Manager software to be used for the current HA setup. We select TSA as shown in Example 12-99.

### *Example 12-99 Select cluster manager software*

---

The following are valid settings for the high availability configuration parameter:

- 1.TSA
- 2.Vendor

Enter a value for the high availability configuration parameter: [1]

Setting a high availability configuration parameter for instance db2inst1 to TSA.

---

## 5. Choose a failover policy.

Now you need to configure the failover policy for the instance db2inst1. The failover policy determines how to fail over the DB2 instance if the active nodes fail.

In this example, we select Active/Passive policy.

db2haicu then prompts for any non-critical mount points to be designated. For this case, we chose to designate only one non-critical mount point.

**Note:** You should add to the non-critical path list all mount points that you are sure you never wish to fail over. This list should include any mount points specified in /etc/fstab that are local mount points and can never be failed over.

We choose the Active/Passive failover policy and specify the host names of an active/passive pair. After that, db2haicu adds DB2 partition0 to the cluster. Example 12-100 shows all the sequences of choosing the failover policy.

### *Example 12-100 Choose failover policy*

---

Now you need to configure the failover policy for the instance db2inst1. The failover policy determines the machines on which the cluster manager will restart the database manager if the database manager is brought offline unexpectedly.

The following are the available failover policies:

1. Local Restart -- during failover, the database manager will restart in place on the local machine
2. Round Robin -- during failover, the database manager will restart on any machine in the cluster domain
3. Active/Passive -- during failover, the database manager will restart on a specific machine
4. M+N -- during failover, the database partitions on one machine will failover to any other machine in the cluster domain (used with DPF instances)
5. Custom -- during failover, the database manager will restart on a machine from a user-specified list

Enter your selection:

**3**

You can identify mount points that are noncritical for failover. For more information, see the topic 'Identifying mount points that are noncritical for failover' in the DB2 Information Center. Are there any mount points that you want to designate as noncritical? [2]

1. Yes
2. No

**1**

Enter the full path of the mount to be made non-critical:

/

Adding path / to the non-critical path list ...

Adding path / to the non-critical path list was successful.

Do you want to add more paths to the non-critical path list? [1]

1. Yes

2. No

**2**

Active/Passive failover policy was chosen. You need to specify the host names of an active/passive pair.

Enter the host name for the active cluster node:

**mensa**

Enter the host name for the passive cluster node:

**lepus**

Adding DB2 database partition 0 to the cluster ...

Adding DB2 database partition 0 to the cluster was successful.

---

## 6. Set up the virtual IP address.

After we add the database partition to the cluster, db2haicu prompts us to create a virtual IP address. In this case, we use 9.43.86.252 as the virtual IP address.

A virtual IP address is a highly available IP address, which fails over between cluster nodes. All clients connecting to database server should use this virtual IP address, so the clients can still connect to the database server in case failover happens.

**Note:** The virtual IP address must not be used in the subnet, and the network mask should be as same as the public network that you want to use.

Example 12-101 shows the screen output in our configuration.

### *Example 12-101 Configure a virtual IP address*

---

Do you want to configure a virtual IP address for the DB2 partition: 0? [2]

1. Yes

2. No

**1**

Enter the virtual IP address:

**9.43.86.252**

Enter the subnet mask for the virtual IP address 9.43.86.252: [255.255.255.0]

**255.255.252.0**

Select the network for the virtual IP 9.43.86.252:

1. db2\_public\_network\_0

Enter selection:

**1**

Adding virtual IP address 9.43.86.252 to the domain ...

Adding virtual IP address 9.43.86.252 to the domain was successful.

All cluster configurations have been completed successfully. db2haicu exiting ...

---

Congratulations! You have successfully added the DB2 instance db2inst1 to the cluster.

Now you can use the SA MP command **lssam** or **db2pd -ha** to check the cluster domain information. Example 12-102 shows the output of these commands. We have one resource group containing three resources, which is online on the active node mensa.

*Example 12-102 Check HA cluster domain information*

```
db2inst1@mensa:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Offline IBM.Application:db2_db2inst1_0-rs:lepup
    '- Online IBM.Application:db2_db2inst1_0-rs:mensa
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
    '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa

db2inst1@mensa:~> db2pd -ha
DB2 HA Status
Instance Information:
Instance Name           = db2inst1
Number Of Domains       = 1
Number Of RGs for instance = 1

Domain Information:
Domain Name              = hotstdby
Cluster Version          = 2.5.0.2
Cluster State            = Online
Number of nodes          = 2

Node Information:
Node Name                State
-----
mensa                    Online
lepup                    Online

Resource Group Information:
Resource Group Name      = db2_db2inst1_0-rg
Resource Group LockState = Unlocked
Resource Group OpState   = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 3
Number of Allowed Nodes  = 2
  Allowed Nodes
  -----
  mensa
  lepup

Member Resource Information:
Resource Name            = db2mnt-home_db2inst1_db2home-rs
Resource State           = Online
Resource Type            = Mount
Mount Resource Path      = /home/db2inst1/db2home
Number of Allowed Nodes  = 2
  Allowed Nodes
```

```

-----
mensa
lepus

Resource Name           = db2_db2inst1_0-rs
Resource State          = Online
Resource Type           = DB2 Partition
DB2 Partition Number    = 0
Number of Allowed Nodes = 2
  Allowed Nodes
-----
mensa
lepus

Resource Name           = db2ip_9_43_86_252-rs
Resource State          = Online
Resource Type           = IP

Network Information:
Network Name            Number of Adapters
-----
db2_public_network_0   2

Node Name               Adapter Name
-----
mensa.itsosj.sanjose.ibm.com eth0
lepus.itsosj.sanjose.ibm.com eth0

Quorum Information:
Quorum Name             Quorum State
-----
db2_Quorum_Network_9_43_86_90:15_14_49 Online
Fail                    Offline
Operator                 Offline

```

---

### ***DB2 operations that add or remove cluster resource elements***

DB2 database manager communicates with the cluster manager whenever there is an instance or database change that requires cluster change, so users are freed from having to perform separate cluster operations.

The DB2 operations that automatically add or remove resources from the cluster manager are as follows:

- ▶ Create or drop a database object.

When you create a database, DB2 adds related resources automatically.

Example 12-103 shows that resources are automatically added to the cluster.

#### *Example 12-103 Create a database*

```

db2inst1@mensa:~> db2 "create database test on /home/db2inst1/db2data"
DB20000I The CREATE DATABASE command completed successfully.
db2inst1@mensa:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online

```



```

|- Online IBM.Application:db2_db2inst1_0-rs
   |- Offline IBM.Application:db2_db2inst1_0-rs:lepup
   '- Online IBM.Application:db2_db2inst1_0-rs:mena
|- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
   |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
   '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mena
|- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs
   |-Offline IBM.Application:db2mnt-home_db2inst1_db2data-rs:lepup
   '- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs:mena
'- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
   |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
   '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mena

```

---

When you drop a database, DB2 deletes the related resources automatically too. Example 12-104 shows this behavior.

#### *Example 12-104 Drop a database*

```

db2inst1@mena:~> db2 drop db test
DB20000I The DROP DATABASE command completed successfully.
db2inst1@mena:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
   |- Online IBM.Application:db2_db2inst1_0-rs
   |   |- Offline IBM.Application:db2_db2inst1_0-rs:lepup
   |   '- Online IBM.Application:db2_db2inst1_0-rs:mena
   |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
   |   |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
   |   '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mena
   '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
     |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
     '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mena

```

---

**Note:** When you create or alter table spaces with new file systems, DB2 adds the new file systems to the cluster too. Note that all file systems used must be on the shared storage because we need to fail over them to the other node in case of node failure.

- Change the active log path.

Example 12-105 shows that DB2 adds the new file system `/home/db2inst1/db2log` to the cluster automatically.

#### *Example 12-105 Change active log path*

```

db2inst1@mena:~> db2 "update db cfg for test using newlogpath /home/db2inst1/db2log"
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@mena:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
   |- Online IBM.Application:db2_db2inst1_0-rs
   |   |- Offline IBM.Application:db2_db2inst1_0-rs:lepup

```

```

'- Online IBM.Application:db2_db2inst1_0-rs:mensa
|- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
  |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
  '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
|- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs
  |-Offline IBM.Application:db2mnt-home_db2inst1_db2data-rs:lepup
  '- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs:mensa
|- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
  |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
'- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs
  |- Offline IBM.Application:db2mnt-home_db2inst1_db2log-rs:lepup
  '- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs:mensa

```

---

- Change the database diagnostic log path.

Example 12-106 shows that DB2 adds the new file system /home/db2inst1/db2dump to the cluster automatically.

*Example 12-106 db2diag.log path*

```

db2inst1@mensa:~> db2 "update dbm cfg using diagpath
/home/db2inst1/db2dump"
DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
db2inst1@mensa:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Offline IBM.Application:db2_db2inst1_0-rs:lepup
    '- Online IBM.Application:db2_db2inst1_0-rs:mensa
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
    '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
  |- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs
    |- Offline IBM.Application:db2mnt-home_db2inst1_db2data-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs:mensa
  |- Online IBM.Application:db2mnt-home_db2inst1_db2dump-rs
    |- Offline IBM.Application:db2mnt-home_db2inst1_db2dump-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2dump-rs:mensa
  |- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
  '- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs
    |- Offline IBM.Application:db2mnt-home_db2inst1_db2log-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs:mensa

```

---

- Change the audit log path.

Example 12-107 shows that DB2 adds the new file system /home/db2inst1/db2audit to the cluster automatically.

### Example 12-107 Audit log path

---

```
db2inst1@mena:/home/db2inst1> db2audit configure datapath
/home/db2inst1/db2audit
AUD0000I Operation succeeded.
db2inst1@mena:/home/db2inst1> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Offline IBM.Application:db2_db2inst1_0-rs:lepus
      '- Online IBM.Application:db2_db2inst1_0-rs:mena
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepus
      '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mena
  |- Online IBM.Application:db2mnt-home_db2inst1_db2audit-rs
    |Offline IBM.Application:db2mnt-home_db2inst1_db2audit-rs:lepus
      '-Online IBM.Application:db2mnt-home_db2inst1_db2audit-rs:mena
  |- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2data-rs:lepus
      '- Online IBM.Application:db2mnt-home_db2inst1_db2data-rs:mena
  |- Online IBM.Application:db2mnt-home_db2inst1_db2dump-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2dump-rs:lepus
      '- Online IBM.Application:db2mnt-home_db2inst1_db2dump-rs:mena
  '- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs
    |- Offline IBM.Application:db2mnt-home_db2inst1_db2log-rs:lepus
      '- Online IBM.Application:db2mnt-home_db2inst1_db2log-rs:mena
```

---

For detailed information about the list of paths that are considered to be made HA, consult the IBM DB2 9.5 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/c0052942.html>

## 12.5.3 Testing

Testing is very important when you implement a HA solution. This section describes several test scenarios. For simplification, we only use the resources shown in Example 12-108 during testing.

### Example 12-108 Resources used in testing scenario

---

```
db2inst1@mena:/home/db2inst1> lssam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Offline IBM.Application:db2_db2inst1_0-rs:lepus
      '- Online IBM.Application:db2_db2inst1_0-rs:mena
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepus
      '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mena
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepus
      '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mena
```

---

## Server failure

In this active/passive HA configuration, if the active node fails, all resources fail over to the passive node. If the passive node fails, then all resources stay at the active node. When the passive node come online again, it reintegrates into the cluster.

We simulate server failure by powering off the nodes manually.

Example 12-109 shows that the active node mensa fails over to the passive node lepus successfully. The status of resources on the active node is *Failed Offline*.

### Example 12-109 Active node failure

---

```
lepus:~ # Issam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Online IBM.Application:db2_db2inst1_0-rs:lepus
    '- Failed offline IBM.Application:db2_db2inst1_0-rs:mensa Node=Offline
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:lepus
    '- Failed offline IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa Node=Offline
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepus
    '- Failed offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
Node=Offline
```

---

Now the cluster only runs on the passive node and it is a single point of failure. You need to fix the active node and bring it online as soon as possible to bring the cluster to be a nominal state.

When the active node goes online again, the resources stay at the passive node, and the resource status on the active node changes to *Offline*. Your cluster turns to a nominal state again. Example 12-110 shows the status. If you want to move your database server to the active node, you can use db2haicu to fail back the resource to the active node.

### Example 12-110 Active node backs online after failure

---

```
db2inst1@mensa:~> Issam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Online IBM.Application:db2_db2inst1_0-rs:lepus
    '- Offline IBM.Application:db2_db2inst1_0-rs:mensa
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepus
    '- Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
```

---

Example 12-111 shows the passive node failure. The status of resources on the passive node is *Failed Offline*.

*Example 12-111 Passive node failure*

---

```
db2inst1@mensa:/home/db2inst1> Issam
Online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_0-rs
    |- Failed offline IBM.Application:db2_db2inst1_0-rs:lepus
    '- Online IBM.Application:db2_db2inst1_0-rs:mensa
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Failed offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |- Failed offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepus
    '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
```

---

After the passive node goes online again, the status of resources turns to *Offline* again.

## Network failure

We discuss network failures in this section by simulating network interface malfunctions on the active node. We perform these steps:

1. Unplug the eth0 cable (public network cable) connected to the active node mensa.

As this is a single network topology, the current active node starts to reboot and the resources fail over to the passive node lepus.

2. Issue the **Issam** command to examine the state of the resources. The resources should look like Example 12-109 on page 578.

**Note:** The node that is being rebooted remains offline until the network cable is plugged back in.

To recover from this state, do the following:

1. Plug the eth0 cable back in.
2. Issue the **Issam** or the **db2pd -ha** command repeatedly and observe the system resources. The resources on the active node should turn to *Offline* eventually. Refer to Example 12-110 on page 578 for the status.

The network failure on the passive node should look as same as the passive node failure shown in Example 12-111.

## DB2 instance failure

Here we test for DB2 instance failure by stopping the DB2 instance manually.

Issue the **db2\_ki11** command on the machine where the resource is online.

Run the **lssam** or the **db2pd -ha** command to examine the resources. You see output similar to that illustrated in Example 12-112.

### Example 12-112 Kill DB2 instance

---

```
mensa:~ # lssam
Pending online IBM.ResourceGroup:db2_db2inst1_0-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst1_0-rs
    |- Offline IBM.Application:db2_db2inst1_0-rs:lepup
    '- Pending online IBM.Application:db2_db2inst1_0-rs:mensa
  |- Online IBM.ServiceIP:db2ip_9_43_86_252-rs
    |- Offline IBM.ServiceIP:db2ip_9_43_86_252-rs:lepup
    '- Online IBM.ServiceIP:db2ip_9_43_86_252-rs:mensa
  '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs
    |-Offline IBM.Application:db2mnt-home_db2inst1_db2home-rs:lepup
    '- Online IBM.Application:db2mnt-home_db2inst1_db2home-rs:mensa
```

---

The instance resource is in the *Pending Online* state.

Issue the **lssam** or the **db2pd -ha** command repeatedly and you should see that the cluster manager starts the DB2 instance automatically. This results in the *Pending Online* state changing to the *Online* state.

In some cases, the cluster manager fails to restart the DB2 instance on the active node and initiates a failover operation to the passive node. The process is the same as the active node failure.

## 12.6 DB2 DPF HA configuration with shared disk on AIX

This section describes how to configure automatic failover in a DB2 DPF (Database Partitioning Feature) with TSA environment on AIX using the **db2haicu** command.

In a partitioned database with TSA environment, TSA facilitates failure detections and automatically executes the database takeover from one system to another in the cluster after a hardware or software failure. In a TSA cluster, a partitioned database is placed under TSA control. For more details regarding TSA, refer to Chapter 8, “DB2 with TSA” on page 233.

## 12.6.1 Architecture

Before you set up a partitioned database with a TSA environment, you must plan the cluster environment. Consider the following items:

- ▶ Physical nodes:
  - Decide on the number of physical nodes, DB2 partitions on each node (including catalog partition and data partition), and the node for the NFS server.
  - Confirm software requirements on each node.
- ▶ Networks:
  - Select the network interface that provides client access on each node. This is one of the TSA controlled resources.
  - We recommend that you have separate networks for the DB2 fast communications manager (FCM) of a multi-partitioned database and for the communication between NFS server and clients. FCM and NFS can share one network.
  - Have one network card for the TSA tiebreaker.
- ▶ TSA configuration:

Decide on the cluster domain name. The resources to be included are virtual IP (service IP), DB2 instance, and partitioned database. The **db2haicu** commands automatically create the cluster domain and resource groups.
- ▶ Partitioned database configuration:
  - Design the partitioned database.
  - Decide on the failover policy and the failover node pair.

For detailed information about designing and implementing a partitioned database, refer to *IBM Balanced Warehouse E7100: Design and Implementation* at:

<http://ibm.com/software/data/infosphere/balanced-warehouse/e-class.html>
- ▶ Scripts:

Scripts for starting, stopping, and monitoring DB2. DB2 9.5 automatically install the scripts.

### Lab environment

Figure 12-10 shows the configuration of the partitioned database with TSA in our lab environment.

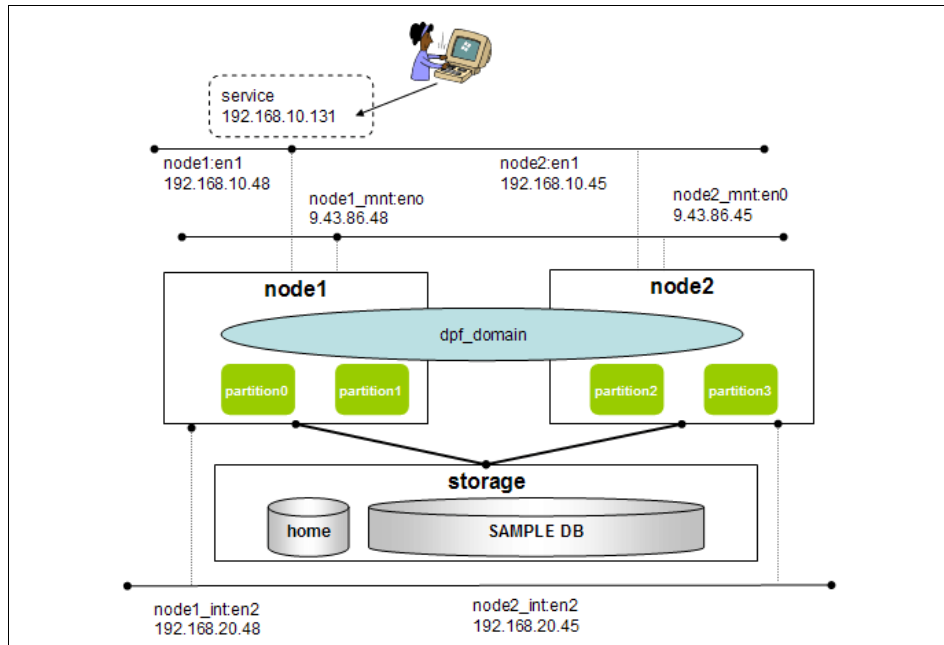


Figure 12-10 multi-partitioned database with TSA lab environment

The planning for our lab environment is as follows:

- ▶ Physical node configuration:
  - Two physical nodes named node1 and node2 are defined in the cluster.
  - The software configuration used to set up the lab environment are:
    - AIX 5.3 Technology Level 7
    - DB2 9.5 Enterprise Server Edition with Fixpack1
    - Tivoli System Automation for Multiplatforms (SA MP) Base Component V2.2
- ▶ Network configuration:
  - One ethernet network interface (en1) on each node is provided for the client's access, which is under the control of TSA. The service address for the clients is added on one of these network interfaces:

**Catalog node (node1)**

**en1: 192.168.10.48 (255.255.252.0)**

**Only data node (node2)**

**en1: 192.168.10.45 (255.255.252.0)**



- One ethernet network interface (en2) is dedicated to the partitioned database internal and NFS client/server communications.

**Catalog node (node1)**

**en2: 192.168.20.48 (255.255.252.0)**

**Only data node (node2)**

**en2: 192.168.20.45 (255.255.252.0)**

- One ethernet network is configured for the TSA tiebreaker. We use en1 for tiebreaker. Refer to Chapter 8, “DB2 with TSA” on page 233 for the description of tiebreaker of TSA in detail.
- ▶ TSA configuration:
  - Cluster Domain named *dpf\_domain* is configured, which includes the resources of the virtual IP (Service IP), the DB2 instance, and the partitioned database.
- ▶ DPF configuration:
  - Each node has the DB2 instance named *db2inst3*.
    - Node1 has two logical partitions (partition0 and partition1). Partition0 is the catalog partition that has system catalog table space and some user data; partition1 is a data partition that has the user data only.
    - Node2 has two logical partitions (partition2 and partition3). Both of these are data partitions containing only the user data.
  - Each instance has the database named *SAMPLE*. The database name should be identical on each node.
  - The failover policy is mutual.
    - If node1 fails, database partition0 and partition1 can be restarted on node2.
    - If node2 fails, database partition2 and partition3 can be restarted on node2.

## 12.6.2 Preparation of the configuration

The following is a summary of the operations required to configure the automated database takeover in a partitioned database environment with TSA:

1. Check the prerequisites
2. Configure the DPF environment
3. Configure the HA partitioned database setup with TSA
4. Perform a joint test of DPF with TSA
  - Normal start up of cluster
  - Planned takeover

- Unplanned takeover
- Normal stop of cluster

The letters in front of a command in the following steps designate on which nodes a command is to be issued to properly set up the topology shown in Figure 12-1 on page 484. The order of the letters also designates the order in which you should issue a command on each node:

- ▶ (A): All nodes.
- ▶ (N1): The node for catalog, NFS server, and data; in our example, node1
- ▶ (N2): The node for data only; in our example, node2

## Hardware and software prerequisites

Information for the hardware and software prerequisites is as follows:

- ▶ The minimum software requirement for running DB2 on AIX can be found at the following Web site:

<http://ibm.com/software/data/db2/9/sysreqs.html>

- ▶ For information about software requirements for running TSA, refer to:

<http://ibm.com/software/tivoli/products/sys-auto-linux/platforms.html>

- ▶ Tivoli System Automation V2.2 manuals can be found at the Web site:

<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>

- ▶ RSCT for AIX manuals can be found at the Web site:

[http://publib.boulder.ibm.com/infocenter/clresctr/vrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct\\_aix5153/b15adm1110.html](http://publib.boulder.ibm.com/infocenter/clresctr/vrx/index.jsp?topic=/com.ibm.cluster.rsct.doc/rsct_aix5153/b15adm1110.html)

- ▶ For information about database partitioning feature (DPF), refer to IBM Balanced Warehouse™ E7100: Design and Implementation at:

<http://ibm.com/software/data/infosphere/balanced-warehouse/e-class.html>

## db2haicu prerequisites

- ▶ The prerequisites specific to db2haicu are covered in:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0051896.html>

## Other prerequisites

Here are a couple of items to be prepared ahead:

- ▶ Synchronize the time and dates.

The time and dates on the primary and the standby nodes should be synchronized as closely as possible. This is absolutely critical to ensure a smooth failover during the primary node failures.

- ▶ Check the shell scripts.

The the start, stop, and monitor scripts provided by DB2 should have already been installed in the directory shown in Example 12-113.

*Example 12-113 DB2 provided scripts*

---

```
(A) #ls -l /usr/sbin/rsct/sapolicies/db2
total 136
-rwxrwxrwx 1 18597 7777 2926 Nov 16 10:57 db2V95_monitor.ksh
-rwxrwxrwx 1 18597 7777 5377 Feb 01 08:42 db2V95_start.ksh
-rwxrwxrwx 1 18597 7777 4555 Feb 01 08:41 db2V95_stop.ksh
-rwxrwxrwx 1 18597 7777 4513 Dec 07 13:11 hadrV95_monitor.ksh
-rwxrwxrwx 1 18597 7777 3621 Aug 27 2007 hadrV95_start.ksh
-rwxrwxrwx 1 18597 7777 3617 Aug 27 2007 hadrV95_stop.ksh
-rwxrwxrwx 1 18597 7777 1312 Aug 27 2007 lockreqprocessed
-rwxrwxrwx 1 18597 7777 4615 Feb 13 09:12 mountV95_monitor.ksh
-rwxrwxrwx 1 18597 7777 4439 Feb 13 09:12 mountV95_start.ksh
-rwxrwxrwx 1 18597 7777 5872 Feb 13 09:12 mountV95_stop.ksh
```

---

If you do not have the scripts, execute the **db2cpts** commands to create the scripts:

```
(A) #mk /usr/sbin/rsct/sapolicies/db2
(A) #/opt/IBM/db2/V9.5/install/tsamp/db2cpts
```

## 12.6.3 Setting up the partitioned database

The following is a summary of steps required to set up the partitioned database:

1. AIX system setup and configuration:
  - a. Create the required operating system groups and users.
  - b. Set applicable kernel parameters.
  - c. Set up the /etc/hosts file.
  - d. Create volume groups, logical volumes, and file systems for the data and the administration partitions. Measure the I/O performance of the system to verify that the storage is set up correctly.
  - e. Install DB2 and TSA software on all nodes.
2. TSA cluster domain setup and configuration:
  - a. Prepare for volume takeover
  - b. Initial configurations of TSA

3. Highly available NFS server configuration:
  - a. Prepare for configuration
  - b. Change NFS server configuration
  - c. TSA configuration about NFS
  - d. Start up HA NFS server
4. DB2 partitioned database setup and configuration:
  - a. Set up passwordless ssh for the DB2 instance owner.
  - b. Create the DB2 instance and set up TCP/IP communications.
  - c. Configure the `/etc/services` files for all servers in the cluster.
  - d. Configure the registry variables and database manager configuration parameters for the DB2 instance.
  - e. Define the database partition servers in the `db2nodes.cfg` file.
  - f. Start the instance and verify that all database partitions are started.
  - g. Create the database.
  - h. Set the database configuration parameters.

We provide the details of the steps in the following sections.

Figure 12-11 shows the entries of the services and hosts files for the partitioned database in our lab environment.

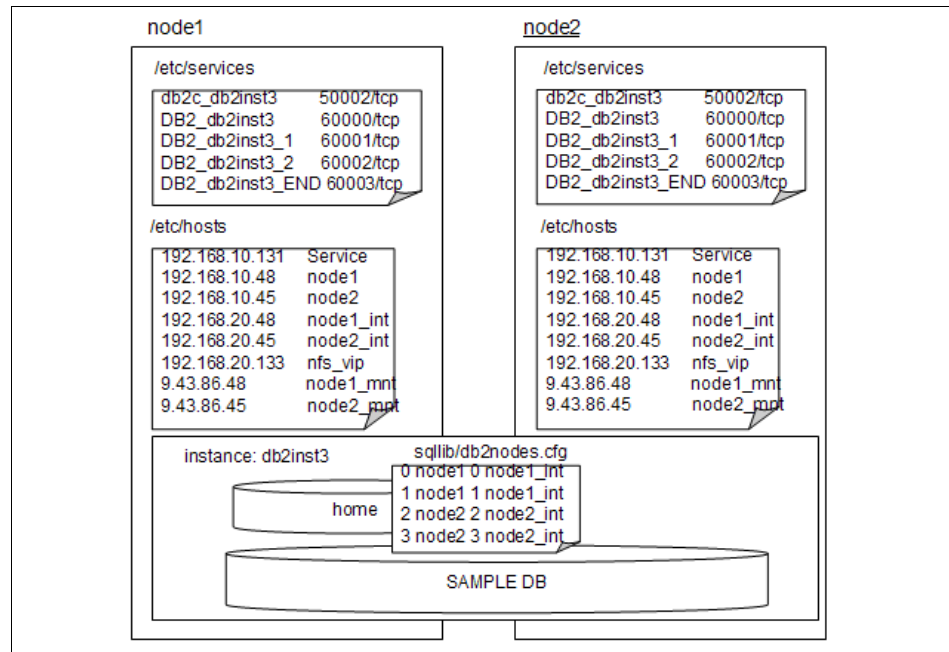


Figure 12-11 Configuration of the partitioned database

## AIX system setup and configurations

This topic provides an overview of the AIX setup and configuration tasks for the partitioned database environment:

1. Create the required operating system groups and users.
  - a. Create three DB2 required user groups.

We create db2iadm1 for the DB2 instance user, db2fadm1 for the DB2 fenced user, and dasadm1 for the DB2 administration server user. These user groups should exist on all nodes in the cluster. It is essential that a group have the same group ID across all nodes.
  - b. Create the three required users.

We create db2inst3 for the DB2 instance owner, db2fenc3 for the DB2 fenced user, and db2das3 for the DB2 administration server user. These users should exist on all nodes in the cluster. It is essential that a user have the same user ID across all nodes.
  - c. To set the password of the all users, use the **passwd** command.
2. Set applicable kernel parameters.

For information about these tasks, see “Chapter 8. Implementation part 4: AIX system setup and configuration” of *IBM Balanced Warehouse E7100: Design and Implementation, SC23-7736-00*.
3. Set up the `/etc/hosts` file.

We include all physical IP addresses and virtual IP addresses on the `/etc/hosts` file. Example 12-114 shows our hosts file.

*Example 12-114 hosts file entries*

---

### (A) #vi /etc/hosts

```
192.168.10.131  Service    ## Virtual IP address for DB2 client/Server
192.168.20.133  nfs_vip    ## Virtual IP address for NFS client/Server

192.168.10.48  node1     ## TSA N/W interface for Virtual IP on node1
192.168.20.48  node1_int ## DPF internal N/W interface on node1
9.43.86.48     node1_mnt ## For Maintenance N/W interface on node1

192.168.10.45  node2     ## TSA N/W interface for Virtual IP on node2
192.168.20.45  node2_int ## DPF internal N/W interface on node2
9.43.86.45     node2_mnt ## For Maintenance N/W interface on node2
```

---

4. Create volume groups for all nodes using these commands as examples:

```
(N1) #mkvg -f -s64 -y vg4 -n -V 49 hdisk4
(N1) #mkvg -f -s64 -y vg5 -n -V 50 hdisk5
(N1) #mkvg -f -s64 -y vg6 -n -V 51 hdisk6
(N2) #mkvg -f -s64 -y vg7 -n -V 52 hdisk7
(N2) #mkvg -f -s64 -y vg8 -n -V 53 hdisk8
```

5. Create the logical volumes and file systems for all nodes. Table 12-5 summarizes the file systems we create in our lab systems for the instance home, database path, and logs on the all nodes.

*Table 12-5 All of the logical volumes and file systems*

Volume group	Logical volume	File system	Function
vg4	/dev/home1v03	/db2home3	DB2 instance file system. Created on node1 only and NFS mounted to all other nodes.
vg5	/dev/dpflv00	/database/db2inst3/NODE0000	File for database path and logs.
vg6	/dev/dpflv01	/database/db2inst3/NODE0001	File for database path and logs.
vg7	/dev/dpflv02	/database/db2inst3/NODE0002	File for database path and logs.
vg8	/dev/dpflv03	/database/db2inst3/NODE0003	File for database path and logs.

These steps describe how to create the file system on a single nodes. The example assumes that the volume group is vg4 and the physical volumes hdisk4.

- a. Create the AIX logical volume /dev/home1v03 with size 1 GB (16 PPs):

```
(N1)#mklv -t jfs2 -y home1v03 vg4 16 hdisk4
(A) #mkdir -p /db2home3
```

- b. Create the AIX file system /db2home3:

```
(N1) #crfs -v jfs2 -d home1v03 -m /db2home3 -a log=INLINE -A no
-p rw
(N1) #mount /db2home3
(N1) #chown -R db2inst3:db2iadm1 /db2home3
```

To create another file system, repeat these steps for every nodes, substituting the appropriate volume groups, physical volumes, logical volumes, and file system names.

6. Install the DB2 Enterprise Edition and TSA software on all nodes.

## **TSA cluster domain configurations**

In this section we discuss the TSA cluster domain setup tasks for a partitioned database environment.

### **Preparation for volume takeover**

The DB2 related volume groups on every nodes should be imported from the other node of the failover pair. The major numbers of all volume groups should be same on both nodes in the pair. To accomplish this, proceed as follows:

1. Determine the major number of the volume groups on all nodes using the following command:

```
ls -al /dev/volumegroupname
```

The volume group major number is equivalent to the major device number of the special device file. For example, in the following example, the major number of the volume group is 51.

```
(N1) #ls -al /dev/vg4
crw-rw----  1 root    system      51,  0 Apr 18 09:43 /dev/vg4
```

2. Import the major number from the owner nodes to the failover nodes.

List the available major number on the failover node use the **lvfstmajor** command:

```
(N2) #lvfstmajor
48....,62...
```

If the major number is available on the failover node, import the volume group, specifying the required major numbers:

```
importvg -y vname -V majornumber disks
(N1) #varyoffvg vg4
(N2) #importvg -y vg4 -V 51 hdisk4
```

If any required major number is not available on the failover node, you must re-configure the volume group to have the major number available.

3. Make sure that all the file systems on all the volume groups just imported do not come online automatically after a reboot.

The **importvg** command updates the device files and the file entries in `/etc/filesystems`. Edit the `/etc/filesystems` file and change the `mount` field to `mount = false` on all nodes.

4. Configure all the volume groups so they are not online automatically after a reboot on failover node:

```
(N2) #chvg -a n vg4
(N2) #varyoffvg vg4
```

Repeat step 1 to 4 on page 589 for all volume groups on all nodes.

## **Initial configurations of TSA**

The initial TSA configuration includes the following steps:

1. Set up SSH for the root user.

Many of the TSA commands used in setup steps require SSH on all nodes. Here we describe how to set up ssh for the root user so that no password is required when commands are entered. For additional information, consult the following article:

<http://ibm.com/developerworks/db2/library/techarticle/dm-0506finnie/>

As a root user, execute the following procedures on all nodes:

- a. If the `/.ssh` directory does not yet exist, create it. Ensure that the `/.ssh` directory does *not* allow group or other write access.

- b. From the `/.ssh` directory, generate a public key and private key pair.

```
(A) #ssh-keygen -t rsa
```

When prompted for input, press Enter. Do not enter a passphrase.

- c. To enable the new key pair for use with ssh, execute the following commands:

```
(A) #mv /.ssh/id_rsa /.ssh/identity
```

```
(A) #chmod 600 /.ssh/identity
```

```
(A) #cat /.ssh/id_rsa.pub >> /.ssh/authorized_keys
```

```
(A) #chmod 644 /.ssh/authorized_keys
```

```
(A) #rm /.ssh/id_rsa.pub
```

- d. Use the **ssh-keyscan** utility to gather the public host key for each host. Substitute the IP address in the example with yours.

```
(A) #ssh-keyscan -t rsa node1_mnt,9.43.86.48,  
node1,192.168.10.48, node1_int,192.168.20.48 >> /.ssh/known_hosts
```

```
(A) #ssh-keyscan -t rsa node2_mnt,9.43.86.45,  
node2,192.168.10.45, node2_int,192.168.20.45 >> /.ssh/known_hosts
```

- e. Test that passwordless ssh is now enabled. On each node, execute the following command:

```
(A) #ssh node2 date
```

Make sure that this command succeeds without prompting you for additional verification.

2. On all nodes, set the environment variable `CT_MANAGEMENT_SCOPE` to 2 (peer domain scope). To permanently set the environment variable, add the following command to the user's root profile file `.profile`:

```
(A) export CT_MANAGEMENT_SCOPE=2
```



3. Run the following command as root to prepare the local node for joining the domain:

```
preprnode nodename1 nodename2
```

```
(A) # preprnode node1 node2
```

4. Create the cluster domain.

Create the cluster domain using the **mkrpdomain** command from either node.

```
mkrpdomain domainname nodename1 nodename2
```

```
(N1) # mkrpdomain dpf_domain node1 node2
```

Now start the cluster domain on either node as follows:

```
startdomain domainname
```

```
(N1) # startdomain dpf_domain
```

Verify that the **dpf\_domain** is online using the **lsrpdomain** command:

```
(A) # lsrpdomain
```

Name	OpState	RSCTActiveVersion	MixedVersions	TSPort	GSPort
dpf_domain	Online	2.4.8.3	No	12347	12348

5. Install the TSA sample policies.

TSA requires three scripts to operate HA resources. The monitor script, the start script, and the stop script. You can download these sample scripts from IBM Web site. Note that these scripts are “AS IS” and support is not available.

<http://catalog.lotus.com/wps/portal/topal/results?catalog.c=TivoliSystemAutomation&catalog.catalogName=Tivoli+OPAL&catalog.searchTerms=>

We download the “Automation Policies for AIX” and install it with the **installp** command or **smit** on each node where NFS server runs. The scripts should be in the **nfsserver** sub directory under **/usr/sbin/rsct/sapolicies**.

```
(A) root@node1 # ls -l /usr/sbin/rsct/sapolicies |grep nfs
drwxr-xr-x  2 bin      bin          4096 Apr 17 23:43 nfsserver
```

### **Configuring a highly available NFS server**

In this section we explain how to configure a NFS server to be highly available for failover in case of the NFS server failure.

The **/home/db2inst3** file system contains the home directories for the DB2 users, including the instance owner. Access to the instance owner’s home directory is a requirement for a partitioned database environment. So we use NFS to share the instance owner’s home directory.

## Preparing for configuration

A small file system about 10 MB on a shared disk is required in the failover process for the NFS state data. Create the file system as shown in Example 12-115.

*Example 12-115 Create file system for NFS state data*

---

```
(N1) root@node1 # mkiv -t jfs2 -y lvnfsctrl vg4 1
lvnfsctrl
```

```
(N1) root@node1 # crfs -v jfs2 -d lvnfsctrl -m /nfsctrl -A no -a
log=INLINE
```

```
File system created successfully.
259884 kilobytes total disk space.
New File System size is 524288
```

```
(N1) root@node1 # mount /nfsctrl
```

```
(N1) root@node1 # cat /etc/filesystems
```

```
...
/nfsctrl:
    dev           = /dev/lvnfsctrl
    vfs           = jfs2
    log           = INLINE
    mount         = false
    account       = false
```

---

When a new file system or a new logical volume is created on a shared storage, you have to create the device files and add entries to `/etc/filesystems` on each node. This can be accomplished by the `exportvg` and `importvg` commands.

## Configuring the NFS server

Perform these steps to change the NFS server configuration:

1. Configure the NFS state data files.

The NFS uses the `/etc/rmtab` and `/etc/xtab` files to keep track of the state data about the exported directories and the connected clients. Therefore, we have to keep the state data files highly available the same as the NFS server. To do that, we move these files to the directory `/nfsctrl` created on a shared storage for this purpose, then create the symbolic link point from the `/etc` directory to the `/nfsctrl` directory. See Example 12-116.

*Example 12-116 Configure the NFS state data files*

---

```
(N1) root@node1 # cp -p /etc/xtab /nfsctrl
```

```
(N1) root@node1 # cp -p /etc/rmtab /nfsctrl
```

```
(N1) root@node1 # ls -l /nfsctrl
```

```
total 16
drwxr-xr-x  2 root  system      256 Apr 17 23:20 lost+found
```

```

-rw-r--r-- 1 root    system    20 Apr 17 18:01 rmtab
-rw-rw-r-- 1 root    system    81 Apr 17 20:23 xtab
(N1) root@node1 # mv /etc/xtab /etc/xtab.original
(N1) root@node1 # ln -sf /nfsctrl/xtab /etc/xtab
(N1) root@node1 # mv /etc/rmtab /etc/rmtab.original
(N1) root@node1 # ln -sf /nfsctrl/rmtab /etc/rmtab
(N1) root@node1 # ls -l /etc/*tab
-rw-r--r-- 1 root    system    2451 Apr 03 10:00 /etc/bootptab
-rw-r--r-- 1 root    system    3049 Apr 17 20:09 /etc/inittab
lrwxrwxrwx 1 root    system    14 Apr 17 23:28 /etc/rmtab -> /nfsctrl/rmtab
lrwxrwxrwx 1 root    system    13 Apr 17 23:28 /etc/xtab -> /nfsctrl/xtab

```

---

Perform the same operation on the standby node. Example 12-117 shows how to switch the shared disk ownership and move the NFS state data files.

*Example 12-117 Switch the shared disk ownership*

```

(N1) root@node1 # umount /nfsctrl
(N1) root@node1 # varyoffvg vg4

(N2) root@node2 # varyonvg vg4
(N2) root@node2 # lspv |grep vg4
hdisk4          0009cdfa1a553999                vg4          active
(N2) root@node2 # mount /nfsctrl
(N2) root@node2 # ls -l /nfsctrl
total 16
drwxr-xr-x 2 root    system    256 Apr 17 23:20 lost+found
-rw-r--r-- 1 root    system    20 Apr 17 18:01 rmtab
-rw-rw-r-- 1 root    system    81 Apr 17 20:23 xtab
(N2) root@node2 # mv /etc/xtab /etc/xtab.original
(N2) root@node2 # ln -sf /nfsctrl/xtab /etc/xtab
(N2) root@node2 # mv /etc/rmtab /etc/rmtab.original
(N2) root@node2 # ln -sf /nfsctrl/rmtab /etc/rmtab
(N2) root@node2 # ls -l /etc/*tab
-rw-r--r-- 1 root    system    2451 Apr 07 15:21 /etc/bootptab
-rw-r--r-- 1 root    system    3030 Apr 17 20:10 /etc/inittab
lrwxrwxrwx 1 root    system    14 Apr 17 23:29 /etc/rmtab -> /nfsctrl/rmtab
lrwxrwxrwx 1 root    system    13 Apr 17 23:29 /etc/xtab -> /nfsctrl/xtab

```

---

2. Add a service IP address for NFS service.

A highly available IP address is required to provide HA NFS service. This service IP address should be able to be reached from every node of the cluster domain. It also must be on the DB2 FCM network because it is the network used by the NFS server. In addition, this service IP address must be a TSA resource.

In this scenario, we use “192.168.20.133” for the HA NFS service and add one service IP address in `/etc/hosts` as follows:

```
(A) # cat /etc/hosts
...
192.168.20.133  nfs_vip
```

### 3. Check the NFS server configuration.

All the NFS mounted file systems must have the same configuration on all the nodes where the NFS server can run. Check the following items:

#### – NFS-exported file systems:

The NFS server must be able to mount the NFS-exported file systems on each node. The NFS-exported file systems should be included in the `/etc/filesystems`. These entries should exist in both nodes. If you have already executed `exportvg` and `importvg` commands, the `/etc/filesystems` file should have entries similar to those shown in Example 12-118.

#### *Example 12-118 Entries in /etc/filesystems*

---

```
(A) root@node1 # ls -l /dev |grep home1v03
brw-rw---- 1 root  system    51, 2 Apr 19 21:11 home1v03
crw-rw---- 1 root  system    51, 2 Apr 19 21:11 rhome1v03
(A) root@node1 # cat /etc/filesystems
...
/db2home3:
    dev           = /dev/home1v03
    vfs           = jfs2
    log           = /dev/vg4jfs2log
    mount         = false
    check         = false
    options       = rw
    account       = false
```

---

#### – File system access control:

The `/etc/exports` file contains the access control list for the NFS server. This file defines which file system can be accessed from the NFS clients. Each node must have the same entries in the `/etc/exports` file. Add an entry in the `/etc/exports` file for the home directory of the partitioned database instance:

```
(A) root@node1 # cat /etc/exports
/db2home3
-sec=sys:krb5p:krb5i:krb5:dh,rw,root=nfs_vip:node1:node2:node1_in
t:node2_int:node1_pub:node2_pub
```

**Note:** We add the `nfs_vip` host name in the `root` keyword. It means that the host name, `nfs_vip`, needs root access authority. Because the NFS client on the owner node can access the NFS server through the service IP address, we need to add a label of the service IP address to a `root` keyword.

- Disable the automatic start at the system boot time.

TSA automatically starts the required services to run the NFS server. Therefore, these services must be turned off on all the nodes where the NFS server can run. When the NFS resource group fails on the owner node, TSA tries to start the NFS services on the failover node to avoid any downtime. If the NFS services start automatically at the boot time, the failed node attempts to restart another NFS server when it is restored, even though the NFS server is already running on the new owner node. To prevent this, ensure that the NFS server does not automatically start at the boot time on either node of the cluster domain. You can do this by turning off NFS using `smit`:

Enter the command `smit nfs` → **Select Network File System (NFS)** → **Select Configure NFS on This System** → **Select Stop NFS**

Select “**both**” on the prompt asking to choose from STOP NFS “now”, “system restart” or “both”,

4. Configure the NFS client.

Edit the `/etc/filesystems` file to include an entry for the NFS-mounted home directories:

- Change the `nodename` keyword to the host name of the service IP address for NFS in the `/etc/filesystems` file.
- Change the `mount` keyword to `true` so this NFS-mounted home directory is automatically mounted when the nodes are started.

Change the entries as shown in the Example 12-119.

*Example 12-119 Output of `/etc/hosts` and `/etc/filesystems`*

---

```
(A) root@node1 # cat /etc/hosts
...
192.168.20.133 nfs_vip
(A) root@node1 # cat /etc/filesystems
...
/home/db2inst3:
    dev           = "/db2home3"
    vfs           = nfs
    nodename      = nfs_vip
    mount         = true
```

```

options          =
bg,soft,intr,rsize=8192,wsiz=8192,sec=sys:dh:krb5:krb5i:krb5p
account          = false
...

```

---

5. Verify the following information:

- The volume group for NFS-exported file system is not “varied on” on either node that can be the NFS server.
- The volume group on either node is not set up to vary on automatically when the machine starts.
- The major numbers of the volume groups are the same on both nodes.
- No NFS-exported file system is mounted on both nodes.

### TSA configuration for NFS

In this section we create the TSA configuration for the HA NFS service.

1. Edit the sample policy configuration file.

The *sa-nfssserver.conf* file in the */usr/sbin/rsct/sapolicies/nfssserver* directory is for setting up the highly available NFS server. Change this configuration file for your environment on all nodes. Example 12-120 highlights in *italic* the lines that should be changed in *sa-nfssserver.conf*.

*Example 12-120 Customizing the sa-nfssserver.conf file*

---

```

##### START OF CUSTOMIZABLE AREA #####
#
# set default values

# --directory for control scripts
script_dir="/usr/sbin/rsct/sapolicies/nfssserver"

# --prefix of all NFS server resources
prefix="SA-nfssserver-"

# --list of nodes in the NFS server cluster
nodes="node1 node2"

# --IP address and netmask for NFS server,
# If more instances of <ip_>, add more rows, like: ip_2 ip_3...
ip_1="192.168.20.133,255.255.252.0"

# --List of network interfaces ServiceIP ip_x depends on.
# Entries are lists of the form <network-interface-name>:<node-name>,...
# If more instances of <nieq_>, add more rows, like: nieq_2 nieq_3...
# Default: create 'static equivalencies'
# to create 'dynamic equivalencies' use keyword nieq_1_dyn ...
nieq_1="en2:node1,en2:node2"

```

```

# --common local mountpoint for shared data
# If more instances of <data_>, add more rows, like: data_tmp, data_proj.
# Note: the keywords need to be unique!
data_varlibnfs="/nfsctrl"
data_work="/db2home3"

# --LVM definitions: VG and optional hdisk ( for AIX only )
# one entry allowed, like: myvg ... with hdisk like: myvg hdisk5
lvm="vg4 hdisk4"

```

---

## 2. Run the NFS server configuration script.

On one of the nodes, execute the configuration script `cfgnfsserver` to create a resource group and all resources of the NFS server. You should execute this script on the `/usr/sbin/rsct/sapolicies/nfsserver` directory because it accesses to the other library by relative path. Example 12-121 shows the result of the script execution.

*Example 12-121 Sample output of the configuration script*

---

```
(N1) root@node2 # cd /usr/sbin/rsct/sapolicies/nfsserver
```

```
(N1) root@node2 # ./cfgnfsserver -p
```

```
using option: 'perform commands!'.
```

Attention:

If resources matching 'SA-nfsserver-\*' exist, they will be removed.

Continue y|n?

y

```

successfully performed: 'mkrgr SA-nfsserver-rg'
Generated definitions: 'SA-nfsserver-server.def'
successfully performed: 'mkrsrc -f SA-nfsserver-server.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-nfsserver-rg IBM.Application:SA-nfsserver-server'
Generated definitions: 'SA-nfsserver-ip-1.def'
successfully performed: 'mkrsrc -f SA-nfsserver-ip-1.def IBM.ServiceIP'
successfully performed: 'addrgmbr -m T -g SA-nfsserver-rg IBM.ServiceIP:SA-nfsserver-ip-1'
successfully performed: 'mkrel -S IBM.Application:SA-nfsserver-server -G
IBM.ServiceIP:SA-nfsserver-ip-1 -p DependsOn SA-nfsserver-server-on-ip-1'
successfully performed: 'mkequ SA-nfsserver-nieq-1 IBM.NetworkInterface:en2:node2,en2:node1'
successfully performed: 'mkrel -S IBM.ServiceIP:SA-nfsserver-ip-1 -G
IBM.Equivalency:SA-nfsserver-nieq-1 -p DependsOn SA-nfsserver-ip-on-nieq-1'
Generated definitions: 'SA-nfsserver-lvm.def'
successfully performed: 'mkrsrc -f SA-nfsserver-lvm.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-nfsserver-rg IBM.Application:SA-nfsserver-lvm'
Generated definitions: 'SA-nfsserver-data-nfsctrl.def'
successfully performed: 'mkrsrc -f SA-nfsserver-data-nfsctrl.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-nfsserver-rg
IBM.Application:SA-nfsserver-data-nfsctrl'
successfully performed: 'mkrel -S IBM.Application:SA-nfsserver-server -G
IBM.Application:SA-nfsserver-data-nfsctrl -p DependsOn SA-nfsserver-server-on-data-nfsctrl'
successfully performed: 'mkrel -S IBM.Application:SA-nfsserver-data-nfsctrl -G
IBM.Application:SA-nfsserver-lvm -p DependsOn SA-nfsserver-data-nfsctrl-on-lvm'
Generated definitions: 'SA-nfsserver-data-work.def'
successfully performed: 'mkrsrc -f SA-nfsserver-data-work.def IBM.Application'
successfully performed: 'addrgmbr -m T -g SA-nfsserver-rg IBM.Application:SA-nfsserver-data-work'
successfully performed: 'mkrel -S IBM.Application:SA-nfsserver-server -G
IBM.Application:SA-nfsserver-data-work -p DependsOn SA-nfsserver-server-on-data-work'

```

```
successfully performed: 'mkrel -S IBM.Application:SA-nfssserver-data-work -G
IBM.Application:SA-nfssserver-lvm -p DependsOn SA-nfssserver-data-work-on-lvm'
```

Resource group, resources and relationships were removed and created.

---

### 3. Change equivalencies type from static to dynamic.

The **cfgnfssserver** script creates the “static” type of equivalencies for the network interfaces of each node. This type of equivalency has some disadvantages. TSA removes the network interfaces from the equivalency when they are deleted from the cluster definition. TSA deletes a network interface from the cluster definition if its IP address is removed, or if the network interface is detached. Moreover, TSA does not add the network interfaces to the equivalency again if the NICs are detected later. As a result, the member of equivalency member is reduced in these cases.

To avoid this, we recommend that you change the definition of equivalency from static to dynamic using the following command:

```
(N1) root@node2 # chequ -u r -D 'Name like "en2" && (NodeNameList={"node1"}
OR NodeNameList={"node2"})' SA-nfssserver-nieq-1 IBM.NetworkInterface
```

This command should be in a single line. The previous example is split into two lines due the limitation of the page width.

### 4. Verify the equivalency configuration.

In the output of the **lsequ -Ab** command, the *SelectString* field should have the values set in the previous step.

```
(N1) root@node2 # lsequ -Ab
Displaying Equivalency information:
All Attributes
Equivalency 1:
      Name = SA-nfssserver-nieq-1
      MemberClass = IBM.NetworkInterface
      Resource:Node[Membership] = {}
      SelectString = "Name like "en2" &&
(NodeNameList={"node1"} OR NodeNameList={"node2"})"
      SelectFromPolicy = ANY
      MinimumNecessary = 1
      Subscription = {}
      ActivePeerDomain = DPF_DOMAIN
      Resource:Node[ValidSelectResources] = {en2:node2,en2:node1}
      Resource:Node[InvalidResources] = {}
      ConfigValidity =
      AutomationDetails[CompoundState] = Undefined
...

```



## Start the HA NFS server

Start the HA NFS server by using the following commands and confirm that the resource group “SA-nfsserver-rg” is online:

```
(N1) # chrg -o online SA-nfsserver-rg
```

Example 12-122 shows that SA-nfsserver-rg is online.

### *Example 12-122 NFS server started*

---

```
(N1) root@node2 # chrg -o online SA-nfsserver-rg
(N1) root@node2 # lssam
Online IBM.ResourceGroup:SA-nfsserver-rg Nominal=Online
  |- Online IBM.Application:SA-nfsserver-data-nfsctrl
    |- Offline IBM.Application:SA-nfsserver-data-nfsctrl:node1
    '- Online IBM.Application:SA-nfsserver-data-nfsctrl:node2
  |- Online IBM.Application:SA-nfsserver-data-work
    |- Offline IBM.Application:SA-nfsserver-data-work:node1
    '- Online IBM.Application:SA-nfsserver-data-work:node2
  |- Online IBM.ServiceIP:SA-nfsserver-ip-1
    |- Offline IBM.ServiceIP:SA-nfsserver-ip-1:node1
    '- Online IBM.ServiceIP:SA-nfsserver-ip-1:node2
  |- Online IBM.Application:SA-nfsserver-lvm
    |- Offline IBM.Application:SA-nfsserver-lvm:node1
    '- Online IBM.Application:SA-nfsserver-lvm:node2
  '- Online IBM.Application:SA-nfsserver-server
    |- Offline IBM.Application:SA-nfsserver-server:node1
    '- Online IBM.Application:SA-nfsserver-server:node2
...

```

---

## Verify the TSA configuration

To verify that all the resource group and resources are created successfully, use the `/usr/sbin/rsct/sapolicies/bin/getstatus` script to list the resources and check if the following resources are created on each node:

- ▶ One for the NFS server (SA-nfsserver-server)
- ▶ One for the service IP address (SA-nfsserver-ip-1)
- ▶ One for the LVM (SA-nfsserver-lvm)
- ▶ One for the mount point resources (SA-nfsserver-data-nfsctrl, SA-nfsserver-data-work)

Example 12-123 shows the resource we created.

### *Example 12-123 Resource created*

---

```
(N1) root@node2 # cd /usr/sbin/rsct/sapolicies/nfsserver
(N1) root@node2 # ../bin/getstatus
```

```
-- Resource Groups and Resources --
```

```
Group Name          Resources
```

```

-----
SA-nfssserver-rg SA-nfssserver-data-nfsctr1
SA-nfssserver-rg SA-nfssserver-data-work
SA-nfssserver-rg SA-nfssserver-ip-1
SA-nfssserver-rg SA-nfssserver-lvm
SA-nfssserver-rg SA-nfssserver-server
-
...
-- Resources --

Resource Name      Node Name      State
-----
SA-nfssserver-data-nfsctr1  node1      Offline
SA-nfssserver-data-nfsctr1  node2      Online
-
SA-nfssserver-data-work    node1      Offline
SA-nfssserver-data-work    node2      Online
-
SA-nfssserver-ip-1        node1      Offline
SA-nfssserver-ip-1        node2      Online
-
SA-nfssserver-lvm         node1      Offline
SA-nfssserver-lvm         node2      Online
-
SA-nfssserver-server      node1      Offline
SA-nfssserver-server      node2      Online
-
...

```

---

## Mount the NFS file systems

Mount the NFS file system on each node:

```
(A) root@node1 # mount /home/db2inst3
```

```
(A) root@node1 # df
```

```

Filesystem      1024-blocks      Free %Used      Iused %Iused Mounted on
...
nfs_vip:/db2home3  5767168  5370352   7%    437    1%
/home/db2inst3

```

This manual mount step is only required during the initial setup. The file system is automated during the failover.

## Simple failover test

To ensure that TSA can move the HA NFS server to the other node and all the NFS clients can access the NFS file system continuously, we perform the following command to move the HA NFS resource group:

```
(N1) # rgrreq -o move SA-nfssserver-rg
```

Example 12-124 shows that TSA has moved the NFS Server from node2 to node1, and node2 can access the /home/db2inst3 directory after failover as an NFS client.

```

(N1) root@node2 # rgreq -o move SA-nfssserver-rg
Action on resource group "SA-nfssserver-rg" returned Token
"0x4e56158c7777b3e4480903b5000cabcl" .
(N1) root@node2 # lssam
Pending online IBM.ResourceGroup:SA-nfssserver-rg Request=Move Nominal=Online
|- Online IBM.Application:SA-nfssserver-data-nfsctrl
|   |- Online IBM.Application:SA-nfssserver-data-nfsctrl:node1
|   '- Offline IBM.Application:SA-nfssserver-data-nfsctrl:node2
|- Online IBM.Application:SA-nfssserver-data-work
|   |- Online IBM.Application:SA-nfssserver-data-work:node1
|   '- Offline IBM.Application:SA-nfssserver-data-work:node2
|- Online IBM.ServiceIP:SA-nfssserver-ip-1
|   |- Online IBM.ServiceIP:SA-nfssserver-ip-1:node1
|   '- Offline IBM.ServiceIP:SA-nfssserver-ip-1:node2
|- Online IBM.Application:SA-nfssserver-lvm
|   |- Online IBM.Application:SA-nfssserver-lvm:node1
|   '- Offline IBM.Application:SA-nfssserver-lvm:node2
'- Pending online IBM.Application:SA-nfssserver-server
|   |- Pending online IBM.Application:SA-nfssserver-server:node1
|   '- Offline IBM.Application:SA-nfssserver-server:node2
...
(A) root@node2 # ls -l /home/db2inst3 |grep test
(A) root@node2 # touch /home/db2inst3/test
(A) root@node2 # ls -l /home/db2inst3 |grep test
-rw-rw-rw- 1 nobody  nobody          0 Apr 18 13:25 test
(N2) root@node2 # rgreq -o move SA-nfssserver-rg

```

## DB2 configurations

Now that NFS and TSA have been successfully configured and tested on their own, this section provides a high level DB2 setup and configuration procedure for a partitioned database environment. For more information about creating a partitioned database, refer to *Database Partitioning, Table Partitioning, and MDC for DB2 9*, SG24-7467-00.

### Set up passwordless ssh for the DB2 instance

On Linux and UNIX systems, a remote shell utility is required for the partitioned database systems. DB2 supports the RSH or SSH remote shell utilities.

By default, DB2 uses RSH when executing commands on the remote DB2 nodes, for example, starting a remote DB2 database partition. To use the DB2 default, the rsh-server package must be installed.

The RSH utility transmits passwords in clear text over the network, which can be a security exposure if the DB2 server is not on a secure network. You can use the DB2RSHCMD registry variable to set the remote shell program to a more secure alternative to avoid this exposure. SSH is one example of a more secure alternative.

Use the following steps to set up SSH for the DB2 instance user so that no password is required when commands are entered:

1. Log in as the instance owner on either node.
2. NFS must be configured and all servers must have the instance home directory (/home/db2inst3) mounted.
3. If the /home/db2inst3/.ssh directory does not yet exist, create it. Ensure that the /.ssh directory does *not* allow group or other write access.
4. From the /home/db2inst3/.ssh directory, generate public key/private key pair.

```
(N1) $ssh-keygen -t rsa
```

When prompted for input, press Enter. Do not enter a passphrase.

5. To enable the new key pair for use with ssh, execute the following commands:

```
(N1) $mv /home/db2inst3/.ssh/id_rsa /home/db2inst3/.ssh/identity
```

```
(N1) $chmod 600 /home/db2inst3/.ssh/identity
```

```
(N1) $cat /home/db2inst3/.ssh/id_rsa.pub >>
```

```
/home/db2inst3/.ssh/authorized_keys
```

```
(N1) $chmod 644 /home/db2inst3/.ssh/authorized_keys
```

```
(N1) $rm /home/db2inst3/.ssh/id_rsa.pub
```

6. Use the **ssh-keyscan** utility to gather the public host key for each host. Replace the IP addresses in the sample commands with yours.

```
(N1) $ssh-keyscan -t rsa node1_mnt,9.43.86.48, node1_int,192.168.20.48,  
node1,192.168.10.48, nfs_vip,192.168.20.133 >>  
/home/db2inst3/.ssh/known_hosts
```

```
(N1) $ssh-keyscan -t rsa node2_mnt,9.43.86.45, node2_int,192.168.20.45,  
node2,192.168.10.45, nfs_vip,192.168.20.133 >> known_hosts
```

7. Test that the passwordless ssh is now enabled. From the all nodes, execute the following command:

```
(A) $ssh node2 date
```

Make sure that this command succeeds without prompting you for additional verification.

8. Set the DB2 registry variable.

If you choose to use the SSH remote shell utility, you need to set the DB2RSHCMD registry variable after creating the DB2 instance. If this registry variable is not set, rsh is used:

```
(N1) $db2set DB2RSHCMD=ssh
```

For additional information about SSH, consult the following article:

<http://ibm.com/developerworks/db2/library/techarticle/dm-0506finnie/>

## ***Set up a partitioned database***

We use the following steps to create a partitioned database in two physical nodes. Each has two database partitions:

### 1. Create the DB2 instance.

- Create the instance using the **db2icrt** command:

```
(N1) #/opt/IBM/db2/V9.5/instance/db2icrt -u db2fenc3 db2inst3
```

db2icrt has to be run as a root user and requires write access to the DB2 home directory, which is NFS mounted. Make sure that root has write access to the DB2 instance owner home directory before running this command.

- The partitioned database environment uses TCP/IP communications for remote client connections and for sending data between partitions. You should ensure that TCP/IP is enabled for all networks on the partitioned database systems using the command:

```
(N1) #db2set DB2COMM=tcPIP
```

- Update the SVCENAME database manager configuration parameter using the appropriate value listed in the `/etc/services` file. In our scenario the instance name is `db2inst3` and the service name is `db2_db2inst3`, the command would look like this:

```
(N1) #db2 update dbm config using svcename db2c_db2inst3
```

- Disable the fault monitor at the instance level using the command:

```
(N1)# db2fm -i db2inst3 -f no
```

### 2. Configure the `/etc/services` files for all nodes in the cluster.

The communication between the database servers in the partitioned database environment must be enabled. The communication between database partition servers is handled by the Fast Communications Manager (FCM). To enable FCM, a port or port range must be reserved for FCM in the `/etc/services` file on each node in the partitioned database system. The reserved ports has the following format (port numbers can vary):

```
(A) vi /etc/services
DB2_db2inst3      60000/tcp
DB2_db2inst3_1   60001/tcp
DB2_db2inst3_2   60002/tcp
DB2_db2inst3_END 60003/tcp
```

The only mandatory entries are the beginning (`DB2_InstanceName`) and the ending (`DB2_InstanceName_END`) ports. The other entries are reserved in the services file so that other applications do not use these ports.

The entries in the services file should be equal to the total number of partitions in the partitioned database system. Because DB2 also has intra-partitions communication, this setup is required even if you only have one physical node with multiple logical partitions.

**Note:** If the environment includes a high availability (HA) solution that is configured to fail over database partitions from one server to another, the `/etc/services` file must contain all the entries for each database partition that the server could potentially host after a failover occurs. For example, the services file should have six entries if there are two physical nodes with three database partitions in each of them and the one node fails over all partitions to the other node.

3. Define the database partition servers in the `db2nodes.cfg` file.

The mapping of database partitions to servers is specified in the `db2nodes.cfg` file, which is found in the instance home directory (that is, `~/sqllib/db2nodes.cfg`). When an instance is created, DB2 automatically creates the `db2nodes.cfg` file with an entry for the instance-owning database partition server. For example, when we create the DB2 instance on node1, the `db2nodes.cfg` file looks like this:

```
(N1) #cat db2nodes.cfg
0 node1 0
```

The fields are the database partition server number (node number), the TCP/IP host name of the server where the database partition server resides, and the logical port number for the database server partition.

Our example partitioned database (illustrated in Figure 12-10 on page 582) has the following layout:

- There are two physical nodes (databases server).
- There are two logical database partitions in each node.
- Database partition0 has the database catalog and is also the coordinator database partition.
- Database partition1 through partition3 have partitioned data.

We update `db2nodes.cfg` as follows:

```
(N1) #vi db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node2 0 node2_int
3 node2 1 node2_int
```

4. Start the instance and verify that all database partitions are started.
5. Check that all the file systems for database in Table 12-5 on page 588 are mounted.
6. Create the database.  
(N1) db2samp1 -dbpath /database  
(A) \$db2 connect to sample

## 12.6.4 Setting up the cluster domain by db2haicu

In this section, we discuss how to configure a DB2 instance for high availability with TSA. TSA automatically responds to the unexpected failure of a server by moving the database partitions from the failed nodes to its failover pair. Any operations against the databases on that instance are rolled back, but the instance is automatically brought back online.

Make sure that all the nodes to be configured into the TSA cluster domain can see and communicate with each other in the network.

We demonstrate the cluster configuration using db2haicu XML file setup mode. For configuring the TSA cluster with db2haicu interactive setup mode, refer to 12.3, “DB2 HADR configuration for automated failover on Linux” on page 508.

Issue the **db2haicu** command to create the resource group of DB2 on either node:

1. Meet the prerequisites for db2haicu.

The db2haicu prerequisites and how to meet the prerequisites are covered in 12.1, “db2haicu” on page 478.

The DB2 Information Center also has the details:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0051896.html>

**Note:** In a multi-partitioned database environment, you have to set the DB2\_NUM\_FAILOVER\_NODES registry variable to specify the number of additional database partitions to be started on a machine in the event of failover.

In our example, each node has two database partitions, we set DB2\_NUM\_FAILOVER\_NODES to 2 so when a failover occurred, four partitions are started on the failed over node.

```
(N1) #db2set DB2_NUM_FAILOVER_NODES=2
```

In a DB2 high availability environment, if a database server fails, the database partitions on the failed node can be restarted on another machine. The FCM uses DB2\_NUM\_FAILOVER\_NODES to calculate how much memory to reserve on each machine to facilitate this failover. At DB2 start time, FCM reserves enough memory on each node for managing up to the total number of the database partitions so that if one machine fails, the failover node can start the additional partitions.

For more information about DB2\_NUM\_FAILOVER\_NODES, refer to Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.regvars.doc/doc/r0005663.html>

## 2. Create the XML file.

The XML file contains all the information that db2haicu needs to know in order to make a partitioned database instance cooperate with TSA. The sample XML files provided by DB2 are located in the sqllib/samples/ha/xml directory. They also can be found at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0052514.html>

Example 12-125 illustrates the XML file we used. The field descriptions are provided in “Configure cluster domain and resource by db2haicu” on page 488. For more details, refer to:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0053130.html>

*Example 12-125 Sample XML file for partitioned database environment*

```
<?xml version="1.0" encoding="UTF-8"?>
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="db2ha.xsd" clusterManagerName="TSA" version="1.0">
  <ClusterDomain domainName="dpf_domain">
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="192.168.10.51"/>
  </ClusterDomain>
</DB2Cluster>
```



```

    <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
    <Interface interfaceName="en1" clusterNodeName="node1">
    <IPAddress baseAddress="192.168.10.48" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    </Interface>
    <Interface interfaceName="en1" clusterNodeName="node2">
    <IPAddress baseAddress="192.168.10.45" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    </Interface>
    </PhysicalNetwork>
    <PhysicalNetwork physicalNetworkName="db2_private_network_0"
physicalNetworkProtocol="ip">
    <Interface interfaceName="en2" clusterNodeName="node1">
    <IPAddress baseAddress="192.168.20.48" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
    </Interface>
    <Interface interfaceName="en2" clusterNodeName="node2">
    <IPAddress baseAddress="192.168.20.45" subnetMask="255.255.252.0"
networkName="db2_private_network_0"/>
    </Interface>
    </PhysicalNetwork>
    <ClusterNode clusterNodeName="node1"/>
    <ClusterNode clusterNodeName="node2"/>
</ClusterDomain>
<FailoverPolicy>
    <Mutual></Mutual>
</FailoverPolicy>
<DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst3">
    <VirtualIPAddress baseAddress="192.168.10.131" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    <Mount filesystemPath="/database/db2inst3/NODE0000"/>
    <MutualPair systemPairNode1="node1" systemPairNode2="node2" />
    </DB2Partition>
    <DB2Partition dbpartitionnum="1" instanceName="db2inst3">
    <Mount filesystemPath="/database/db2inst3/NODE0001"/>
    <MutualPair systemPairNode1="node1" systemPairNode2="node2" />
    </DB2Partition>
    <DB2Partition dbpartitionnum="2" instanceName="db2inst3">
    <Mount filesystemPath="/database/db2inst3/NODE0002"/>
    <MutualPair systemPairNode1="node2" systemPairNode2="node1" />
    </DB2Partition>
    <DB2Partition dbpartitionnum="3" instanceName="db2inst3">
    <Mount filesystemPath="/database/db2inst3/NODE0003"/>
    <MutualPair systemPairNode1="node2" systemPairNode2="node1" />
    </DB2Partition>
</DB2PartitionSet>
<HADBSet instanceName="db2inst3">
    <HADB databaseName="SAMPLE" />
</HADBSet>
</DB2Cluster>

```

---

3. Configure the system with the **db2haicu** command.

The command syntax is as follows:

```
db2haicu -f XMLfilepath
```

**Note:** Before issuing **db2haicu**, make sure that the NFS files are mounted and the DB2 instance is started on all nodes in the cluster.

Example 12-126 shows the output of **db2haicu** with XML in our test.

*Example 12-126 Sample output of db2haicu with XML*

---

```
(N1) $db2start
(N1) $db2haicu -f db2dpf.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called
db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster
domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic
called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information
Center.

db2haicu determined the current DB2 database manager instance is db2inst3. The cluster
configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as
db2haicu will need to activate all databases for the instance to discover all paths ...
Configuring quorum device for domain dpf_domain ...
Configuring quorum device for domain dpf_domain was successful.
Adding network interface card en1 on cluster node node1 to the network db2_public_network_0 ...
Adding network interface card en1 on cluster node node1 to the network db2_public_network_0 was
successful.
Adding network interface card en1 on cluster node node2 to the network db2_public_network_0 ...
Adding network interface card en1 on cluster node node2 to the network db2_public_network_0 was
successful.
Adding network interface card en2 on cluster node node1 to the network db2_private_network_0 ...
Adding network interface card en2 on cluster node node1 to the network db2_private_network_0 was
successful.
Adding network interface card en2 on cluster node node2 to the network db2_private_network_0 ...
Adding network interface card en2 on cluster node node2 to the network db2_private_network_0 was
successful.
Adding DB2 database partition 0 to the cluster ...
Adding DB2 database partition 0 to the cluster was successful.
Adding DB2 database partition 1 to the cluster ...
Adding DB2 database partition 1 to the cluster was successful.
Adding DB2 database partition 2 to the cluster ...
Adding DB2 database partition 2 to the cluster was successful.
Adding DB2 database partition 3 to the cluster ...
Adding DB2 database partition 3 to the cluster was successful.
Adding database SAMPLE to the cluster domain ...
Adding database SAMPLE to the cluster domain was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
```

---

Issue the **lssam**, **lssrpdomain**, and **lssrpnod** commands to see the resources created during this process. Example 12-127 shows that the cluster nodes, cluster domain, and all the resources are online.

*Example 12-127 Output of Issam, Isrpdomain and Isrpnod commands*

---

**(A) #Issam**

```
Online IBM.ResourceGroup:SA-nfssserver-rg Nominal=Online
|- Online IBM.Application:SA-nfssserver-data-nfsctrl
|'- Offline IBM.Application:SA-nfssserver-data-nfsctrl:node1
|'- Offline IBM.Application:SA-nfssserver-data-nfsctrl:node2
|- Online IBM.Application:SA-nfssserver-data-work
|'- Online IBM.Application:SA-nfssserver-data-work:node1
|'- Offline IBM.Application:SA-nfssserver-data-work:node2
|- Online IBM.ServiceIP:SA-nfssserver-ip-1
|'- Online IBM.ServiceIP:SA-nfssserver-ip-1:node1
|'- Offline IBM.ServiceIP:SA-nfssserver-ip-1:node2
|- Online IBM.Application:SA-nfssserver-lvm
|'- Online IBM.Application:SA-nfssserver-lvm:node1
|'- Offline IBM.Application:SA-nfssserver-lvm:node2
'- Online IBM.Application:SA-nfssserver-server
|'- Online IBM.Application:SA-nfssserver-server:node1
|'- Offline IBM.Application:SA-nfssserver-server:node2
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
|'- Online IBM.Application:db2_db2inst3_0-rs:node1
|'- Offline IBM.Application:db2_db2inst3_0-rs:node2
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
|'- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
|'- Offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
|'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
|'- Offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|'- Offline IBM.Application:db2_db2inst3_1-rs:node1
|'- Online IBM.Application:db2_db2inst3_1-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
|'- Offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
|'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
```

**(A) # Isrpdomain**

Name	OpState	RSCTActiveVersion	MixedVersions	TSPort	GSPort
dpf_domain	Online	2.4.8.3	No	12347	12348

**(A) # Isrpnod**

Name	OpState	RSCTVersion
node1	Online	2.4.8.3
node2	Online	2.4.8.3

---

## Deleting the cluster domain

If you need to delete the cluster domain because of some errors, you can execute the following procedures:

1. Run the **dbhaicu** command with the **-delete** option on either node to delete the HA configuration of DB2.

The **-delete** option removes the entire DB2 HA configuration and deletes all resources in the cluster. If the other instances are using the domain at the time, the domain is deleted as well.

**Note:** The **-delete** option of the **db2haicu** command leaves the DB2 instances unaffected. That is, it does not stop the DB2 instances. However, IP addresses for DB2 that were highly available are removed and no longer present after the **db2hacu -delete** command completes.

After that, you delete the resource group of NFS and the entire configuration.

2. Execute the **db2stop** command on either node to stop the DB2 instance.
3. Execute the **umount** command on each node to unmount the NFS client directory (/home/db2inst3).
4. Run the **chrg** command on either node to stop the resource group of NFS as follows:  
(B) #chrg -o offline SA-nfsserver-rg
5. Run the **rmrpdomain** command on either node to delete the cluster domain with the entire HA configuration.  
(B) #rmrpdomain dpf\_domain

### 12.6.5 Unplanned failure test

The automatic failover function facilitated by TSA provides the high availability for DB2 in the event of unplanned failure. Testing should be conducted after the configuration to ensure that this function is working as expected.

#### DB2 instance failure

We simulate DB2 instance failure by stopping the DB2 processes on node2 as follows:

```
(N2) $db2nkil1
```

From the `lssam` output, we see that the DB2 resource on the node (node2) are in the *Pending Online* state as shown Example 12-128.

*Example 12-128 lssam output after stopping DB2*

---

```
(N1) #lssam
Pending online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst3_2-rs
    |- Offline IBM.Application:db2_db2inst3_2-rs:node1
      '- Pending online IBM.Application:db2_db2inst3_2-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs
    |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node1
      '- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node2
Pending online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
  |- Pending online IBM.Application:db2_db2inst3_3-rs
    |- Offline IBM.Application:db2_db2inst3_3-rs:node1
      '- Pending online IBM.Application:db2_db2inst3_3-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs
    |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node1
      '- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node2
```

---

After that, TSA restarts DB2 partition2 and partition3 on the same node automatically. TSA does not execute takeover if TSA successfully restarts all DB2 partitions. If TSA cannot restart all DB2 partitions, TSA executes a takeover procedure.

### **Non-catalog and NFS client node crash and reintegration**

We simulate the node crash on the DB2 non-catalog and NFS client node (node2) by shutting down the operating system. Power off also achieves this.

#### ***Non-catalog and NFS client node crash and takeover***

When the DB2 non-catalog node (node2) fails, only the data on the failed partitions (partition2 and partition3) cannot be accessed. The data on the alive partitions (partition0 and partition1) is still accessible. The existing transactions on partition0 and partition1 complete, and new connections can be made.

When the NFS client node (node2) fails, TSA does not execute the takeover for the resource group of NFS (SA-nfsserver-rg). To restart only the failed DB2 partitions, TSA executes the takeover for the DB2 resource groups db2\_db2inst3\_2-rg and db2\_db2inst3\_3-rg only from node2 to node1.

Figure 12-12 illustrates the cluster behavior in the event of a node2 crash.

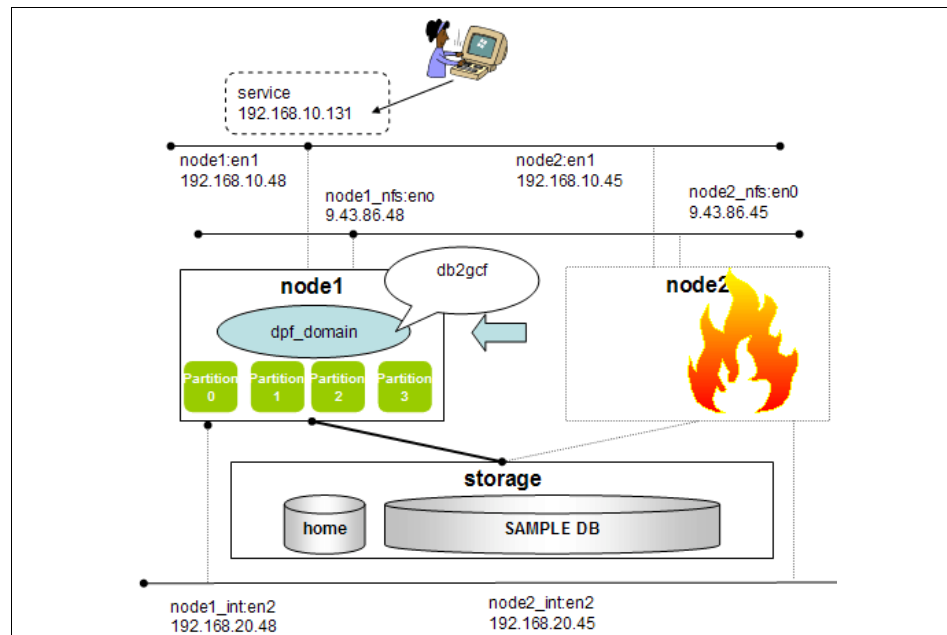


Figure 12-12 Automatic failover to node1

When TSA detects the NFS client node (node2) outage, it starts the failover process:

- ▶ The failover node (node1) acquires the resource group `db2_db2inst3_2-rg` and `db2_db2inst3_3-rg`.
- ▶ TSA invokes the start script on the failover node (node1) to restart DB2 partition2 and partition3.

The scripts include the `db2gcf` command to restart the failed partitions (partition2 and partition3). The `db2gcf` command changes the host name field, logical port number, and the net name of the failed node in the `db2nodes.cfg` file so the failed partitions can be started on the failover node. The before and after entries of `db2nodes.cfg` in our lab systems are as follows:

Before the takeover:

```
(N1) #cat db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node2 0 node2_int
3 node2 1 node2_int
```

After the takeover:

```
(N1) #cat db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node1 2 node1_int
3 node1 3 node1_int
```

You can issue the `lssam` command to examine the state of the resources during the failover process. Example 12-129 shows the resources states after the failover:

- ▶ The DB2 resource groups (db2\_db2inst3\_2-rg and db2\_db2inst3\_3-rg) are *Online* on the new owner node (node1).

The resources of DB2 (db2\_db2inst3\_2-rs, db2mnt-database\_db2inst3\_NODE0002-rs, db2\_db2inst3\_3-rs, and db2mnt-database\_db2inst3\_NODE0003 -rs) are have the *Online* status on node1.

- ▶ The resources on the old owner node (node2) assume the *Failed Offline* state.

After restarting the resource groups of DB2 partitions (db2\_db2inst3\_2-rg and db2\_db2inst3\_3-rg) on node1, DB2 client can access data on all four partitions.

If you set the DB2 client reroute function, the DB2 client automatically reconnects to the database. If you do not set the DB2 client reroute function, you need to reconnect to the database expressly from applications.

*Example 12-129 lssam output after node crash*

---

```
(N1) #lssam
...
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
   |- Online IBM.Application:db2_db2inst3_0-rs:node1
   |- Failed offline IBM.Application:db2_db2inst3_0-rs:node2 Node=Offline
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
   |- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
   |- Failed offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node2 Node=Offline
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
   |- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
   |- Online IBM.Application:db2_db2inst3_1-rs:node1
   |- Failed offline IBM.Application:db2_db2inst3_1-rs:node2 Node=Offline
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
   |- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
   |- Online IBM.Application:db2_db2inst3_2-rs:node1
```

```

'- Failed offline IBM.Application:db2_db2inst3_2-rs:node2 Node=Offline
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs
  |- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node1
  '- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node2
Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_3-rs
    |- Online IBM.Application:db2_db2inst3_3-rs:node1
    '- Failed offline IBM.Application:db2_db2inst3_3-rs:node2 Node=Offline
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs
    |- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node1
    '- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node2
Node=Offline

```

---

## Non-catalog and NFS client node reintegration

As soon as the failed node (node2) comes back up, TSA changes the status of the resources on the failed node (node2) from the *Failed Offline* state to the *Offline* state as Example 12-130.

### Example 12-130 Issam output after node2's start

```

Issam
...
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_0-rs
    |- Online IBM.Application:db2_db2inst3_0-rs:node1
    '- Offline IBM.Application:db2_db2inst3_0-rs:node2
  |- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
    |- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
    '- Offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
    |- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
    '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_1-rs
    |- Online IBM.Application:db2_db2inst3_1-rs:node1
    '- Offline IBM.Application:db2_db2inst3_1-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
    |- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
    '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_2-rs
    |- Online IBM.Application:db2_db2inst3_2-rs:node1
    '- Offline IBM.Application:db2_db2inst3_2-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs
    |- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node1
    '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_3-rs
    |- Online IBM.Application:db2_db2inst3_3-rs:node1
    '- Offline IBM.Application:db2_db2inst3_3-rs:node2
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs
    |- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node1
    '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node2

```

---

When node2 is back, we recommend that you move partition2 and partition3 back to node2 to balance the data load. Refer to “Move DB2 partitions from one node to the other node” on page 622 for details on how to move the partitions.



## Catalog and NFS server node crash and reintegration

We discuss node failure in this section by simulating the node crash on the DB2 catalog and NFS server node (node1).

### *Catalog and NFS server node crash and takeover*

When the DB2 catalog node (node1) fails, even though partition2 and partition3 are still alive, the database connections are terminated due to the DB2 system catalog table space on the failed node (partition0) is not available.

When the NFS server node (node1) fails, TSA executes the takeover for the resource group of NFS (SA-nfsserver-rg).

To restart the failed DB2 partitions and NFS server, TSA executes the takeover for the DB2 resource groups (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg) and the resource group of NFS (SA-nfsserver-rg) to move them from node1 to node2.

Figure 12-13 shows the cluster behavior in the event of the node1 crash.

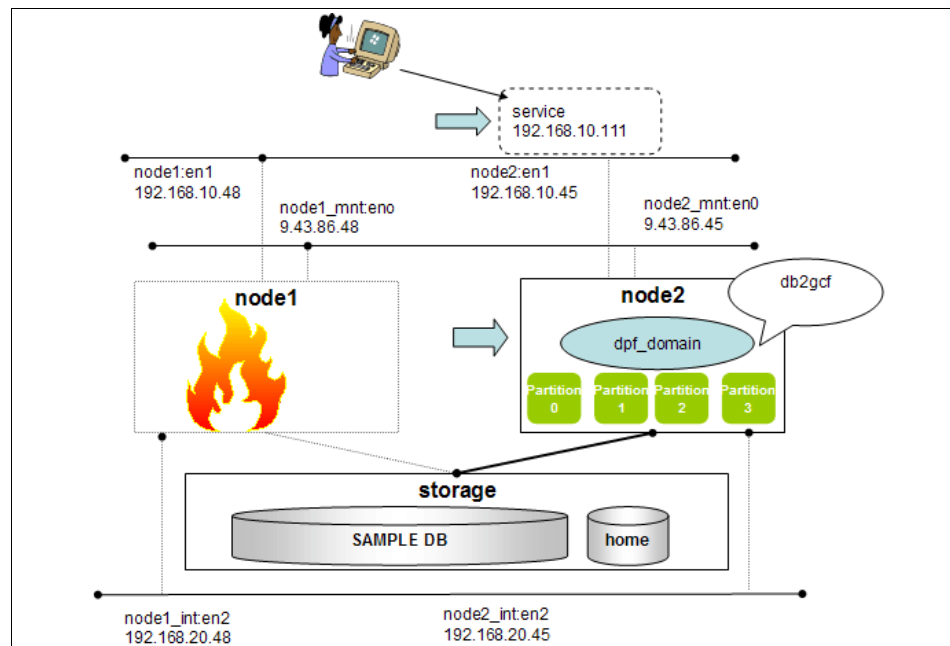


Figure 12-13 Automatic failover to node2

When TSA detects the catalog and NFS server node (node1) crash, it starts the failover process. While executing failover, partition2 and partition3 are alive. However, the DB2 client cannot connect to the database.

- ▶ The failover node (node2) acquires the resource groups of NFS (SA-nfssserver-rg) and DB2 (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg).
- ▶ TSA assigns the virtual IP address (192.168.20.133) for NFS to the en2 NIC on the failover node (node2). The home directory with NFS is also mounted on the failover node.
- ▶ The virtual IP address(192.168.10.131) for DB2 communication between clients and server is assigned to the en1 NIC on the failover node (node2).
- ▶ TSA invokes the start script on the failover node (node2) to restart DB2 partition0 and partition1.

The scripts include the **db2gcf** command to restart the failed partitions (partition0 and partition1). This command changes host name, logical port number, and the netname for FCM for the failed node in the db2nodes.cfg so the failover node can start all the partitions. The before and after content of the db2nodes.cfg of our systems are as follows:

Before the takeover:

```
(N1) #cat db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node2 0 node2_int
3 node2 1 node2_int
```

After the takeover:

```
(N1) #cat db2nodes.cfg
0 node2 2 node2_int
1 node2 3 node2_int
2 node2 0 node2_int
3 node2 1 node2_int
```

You can use the **1ssam** command to examine the state of the resources in the process. After the failover, the resources settle down to the states illustrated in Example 12-131. The resource group of NFS server is *Online* on the new owner node (node2). The DB2 resource groups (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg), the resources of DB2 (db2\_db2inst3\_0-rs, db2mnt-database\_db2inst3\_NODE0000-rs, db2\_db2inst3\_1-rs, and db2mnt-database\_db2inst3\_NODE0001-rs) are also *Online* on node2. The resources on the old owner node (node1) assume the *Failed Offline* state.

After restarting the resource group of DB2 catalog partition (db2\_db2inst3\_0-rg) on node2, the DB2 catalog partition (partition0) can be accessed and the DB2 client can connect to the databases. If you set the DB2 client reroute function, the DB2 client is automatically reconnected to the database. If you do not set the DB2 client reroute function, you need to reconnect to the database explicitly from the applications.

After restarting all the DB2 resource groups on node2, we can access all the data (Example 12-131).

*Example 12-131 Issam output after node crash*

---

```
(N1) #lssam
Online IBM.ResourceGroup:SA-nfssserver-rg Nominal=Online
|- Online IBM.Application:SA-nfssserver-data-nfsctrl
|   |- Failed offline IBM.Application:SA-nfssserver-data-nfsctrl:node1 Node=Offline
|   '- Online IBM.Application:SA-nfssserver-data-nfsctrl:node2
|- Online IBM.Application:SA-nfssserver-data-work
|   |- Failed offline IBM.Application:SA-nfssserver-data-work:node1 Node=Offline
|   '- Online IBM.Application:SA-nfssserver-data-work:node2
|- Online IBM.ServiceIP:SA-nfssserver-ip-1
|   |- Failed offline IBM.ServiceIP:SA-nfssserver-ip-1:node1 Node=Offline
|   '- Online IBM.ServiceIP:SA-nfssserver-ip-1:node2
|- Online IBM.Application:SA-nfssserver-lvm
|   |- Failed offline IBM.Application:SA-nfssserver-lvm:node1 Node=Offline
|   '- Online IBM.Application:SA-nfssserver-lvm:node2
'- Online IBM.Application:SA-nfssserver-server
|   |- Failed offline IBM.Application:SA-nfssserver-server:node1 Node=Offline
|   '- Online IBM.Application:SA-nfssserver-server:node2
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
|   |- Failed offline IBM.Application:db2_db2inst3_0-rs:node1 Node=Offline
|   '- Online IBM.Application:db2_db2inst3_0-rs:node2
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
|   |- Failed offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node1 Node=Offline
|   '- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
|   |- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
|   Node=Offline
|   '- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|   |- Failed offline IBM.Application:db2_db2inst3_1-rs:node1 Node=Offline
|   '- Online IBM.Application:db2_db2inst3_1-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
|   |- Failed offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
|   Node=Offline
|   '- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
...
```

---

### **Catalog and NFS server node reintegration**

As soon as the failed node (node1) comes back, TSA changes the status of the resources on the failed node (node1) from the *Failed Offline* state to the *Offline* state.

In this situation, the four partitions are alive on one server (node2). We recommend that you move the two partitions (partition0 and partition1) back to node1 after node1 is up, to balance the data load.

### **Network of FCM and NFS failure**

In this section we discuss network failures by simulating network interface malfunctions.

## Network of FCM and NFS failure and takeover

In our lab, the en2 is used for the communication between NFS server and clients as well as for DB2 DPF internal communications (FCM). We simulate the private network interface card failure by unplugging the en2 cable going into the NFS server (node1). You also can simulate the network failure for using the **chdev** command:

```
(N1) # chdev -l 'en2' -a state='detach'
```

When TSA detects network outage, it starts the failover process. To restart the failed DB2 partitions and the NFS server, TSA executes the takeover process for the DB2 resource groups (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg) and the resource group of NFS (SA-nfsserver-rg) to move resources from node1 to node2. During the takeover, partition2 and partition3 are alive. However, the DB2 connections are terminated because the DB2 system catalog table space residing on partition0 is not available, as TSA stops partition0 and partition1.

Again, we use the **Issam** command to examine the state of the system resources. Example 12-132 shows the output after unplugging the en2. The status of the resource group of the NFS server (SA-nfsserver-rg) becomes *Pending Offline*. The resource groups of the DB2 partition0 and partition1 (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg) are also in *Pending Online* status.

### Example 12-132 output of Issam after unplug the en2

---

```
(N1)(N2) #Issam
Pending offline IBM.ResourceGroup:SA-nfsserver-rg Nominal=Online
|- Pending offline IBM.Application:SA-nfsserver-data-nfsctrl
|   |- Pending offline IBM.Application:SA-nfsserver-data-nfsctrl:node1
|   '- Offline IBM.Application:SA-nfsserver-data-nfsctrl:node2
|- Pending offline IBM.Application:SA-nfsserver-data-work
|   |- Pending offline IBM.Application:SA-nfsserver-data-work:node1
|   '- Offline IBM.Application:SA-nfsserver-data-work:node2
|- Offline IBM.ServiceIP:SA-nfsserver-ip-1
|   |- Failed offline IBM.ServiceIP:SA-nfsserver-ip-1:node1
|   '- Offline IBM.ServiceIP:SA-nfsserver-ip-1:node2
|- Online IBM.Application:SA-nfsserver-lvm
|   |- Online IBM.Application:SA-nfsserver-lvm:node1
|   '- Offline IBM.Application:SA-nfsserver-lvm:node2
'- Offline IBM.Application:SA-nfsserver-server
|   |- Offline IBM.Application:SA-nfsserver-server:node1
|   '- Offline IBM.Application:SA-nfsserver-server:node2
Pending online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst3_0-rs
|   |- Offline IBM.Application:db2_db2inst3_0-rs:node1
|   '- Pending online IBM.Application:db2_db2inst3_0-rs:node2
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
|   |- Offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
|   '- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
|   |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
|   '- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Pending online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst3_1-rs
|   |- Offline IBM.Application:db2_db2inst3_1-rs:node1
|   '- Pending online IBM.Application:db2_db2inst3_1-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
```

```
| - Offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
|- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
```

---

The same as the node failure, TSA starts the failover process:

- ▶ The failover node (node2) acquires the resource groups of NFS (SA-nfssserver-rg) and DB2 (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg).
- ▶ TSA assigns the virtual IP address (192.168.20.133) for NFS to the en2 NIC on the failover node (node2). The home directory with NFS is also mounted on the failover node.
- ▶ The virtual IP address(192.168.10.131) for DB2 communication between clients and server is assigned to the en1 NIC on the failover node (node2).
- ▶ TSA invokes the start script on the failover node (node2) to restart DB2 partition0 and 1. The scripts include the **db2gcf** command to restart the failed partitions (partition0 and 1).

The **lssam** command can be used to examine the state of the resources in the process. After the failover, the resources settle down to the states illustrated in Example 12-133. The resource group of NFS server and the DB2 resource groups (db2\_db2inst3\_0-rg and db2\_db2inst3\_1-rg) are *Online* on the new owner node (node2).

After restarting the resource group of DB2 catalog partition (db2\_db2inst3\_0-rg) on node2, the DB2 catalog partition (partition0) can be accessed and DB2 client can connect to the databases. If you set the DB2 client reroute function, DB2 clients are automatically reconnected to the database. If you do not set the DB2 client reroute function, you need to reconnect to the database explicitly from the applications.

After restarting all the DB2 resource groups on node2, we can access all the data.

The resource of the virtual IP for NFS (SA-nfssserver-ip-1:node1) remains in *Failed Offline* status on the old owner node (node1).

*Example 12-133 output of lssam after failover*

---

```
(N1)(N2) #lssam
Online IBM.ResourceGroup:SA-nfssserver-rg Nominal=Online
|- Online IBM.Application:SA-nfssserver-data-nfsctrl
|   |- Offline IBM.Application:SA-nfssserver-data-nfsctrl:node1
|   |- Online IBM.Application:SA-nfssserver-data-nfsctrl:node2
|- Online IBM.Application:SA-nfssserver-data-work
|   |- Offline IBM.Application:SA-nfssserver-data-work:node1
|   |- Online IBM.Application:SA-nfssserver-data-work:node2
|- Online IBM.ServiceIP:SA-nfssserver-ip-1
|   |- Failed offline IBM.ServiceIP:SA-nfssserver-ip-1:node1
|   |- Online IBM.ServiceIP:SA-nfssserver-ip-1:node2
|- Online IBM.Application:SA-nfssserver-lvm
|   |- Offline IBM.Application:SA-nfssserver-lvm:node1
```

```

'- Online IBM.Application:SA-nfsserver-lvm:node2
'- Online IBM.Application:SA-nfsserver-server
  |- Offline IBM.Application:SA-nfsserver-server:node1
  '- Online IBM.Application:SA-nfsserver-server:node2
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
  |- Offline IBM.Application:db2_db2inst3_0-rs:node1
  '- Online IBM.Application:db2_db2inst3_0-rs:node2
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
  |- Offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
  '- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
  |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
  |- Offline IBM.Application:db2_db2inst3_1-rs:node1
  '- Online IBM.Application:db2_db2inst3_1-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
  |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2

```

---

### ***Network of FCM and NFS failure reintegration***

To recover from the failure, we plug the en2 cable back into node1. If the simulation is done with the **chdev** command, use the following command to recover the network:

```
(N1) # chdev -l 'en2' -a state='up'
```

After that, the en2 is online and the communications of DB2 FCM and NFS are reestablished. We can access the NFS server and NFS-mounted directory (/home/db2inst3) on the old owner node (node1). For continuity, the DB2 client can remain connected to the database on the new owner node (node2).

Not that the resource of the virtual IP for NFS (SA-nfsserver-ip-1:node1) remains in the *Failed Offline* status on the old owner node (node1).

To remove the Failed Offline flag, run the following commands as the root from one of the nodes to reset the virtual IP resource on the old owner node (node1).

```
resetrsrc -s 'Name like "Virtual IP resource name" &&
NodeNameList={"failed node name"}' IBM.ServiceIP
```

For example:

```
(N1) #resetrsrc -s 'Name like "SA-nfsserver-ip-1" &&
NodeNameList={"node1"}' IBM.ServiceIP
```

From the **Issam** output, you should see the system eventually settle down to the normal state.

## 12.6.6 Planned operations

From time to time, you need to plan a system outage for some maintenance activities such as a software upgrade or to recycle the DB2 instance for changing the non-dynamic database manager parameters. In this section, we demonstrate how to perform some DB2, OS, and TSA operations manually for the planned outage or maintenance.

When you complete the partitioned database with a TSA setup, you can perform these operations with some simple tests to verify your configuration.

### Manual DB2 instance operations

In this section we discuss the situations of stopping and starting DB2 as well as moving database partitions from one node to the other.

#### *Issue the db2stop and db2start command*

Before stopping the DB2 instance, you should gracefully stop the application that connects to DB2. When DB2 is stopped with the **db2stop** command, TSA does not execute the takeover, because it is the planned takeover. You can also use the **db2stop force** command instead of the **db2stop** command, which means that DB2 executes **db2 force applications** and **db2stop** commands in the process of the one command.

All the DB2 resource groups are in *Pending Online* status after DB2 is stopped. See Example 12-134.

#### *Example 12-134 Issam output after stopping DB2*

---

```
(A) #lssam
...
Pending online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst3_0-rs
   |- Offline IBM.Application:db2_db2inst3_0-rs:node1
   '- Offline IBM.Application:db2_db2inst3_0-rs:node2
|- Online IBM.ServiceIP:db2ip_192_168_10_131-rs
   |- Online IBM.ServiceIP:db2ip_192_168_10_131-rs:node1
   '- Offline IBM.ServiceIP:db2ip_192_168_10_131-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node1
   '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0000-rs:node2
Pending online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst3_1-rs
   |- Offline IBM.Application:db2_db2inst3_1-rs:node1
   '- Offline IBM.Application:db2_db2inst3_1-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node1
   '- Offline IBM.Application:db2mnt-database_db2inst3_NODE0001-rs:node2
Pending online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst3_2-rs
   |- Offline IBM.Application:db2_db2inst3_2-rs:node1
   '- Offline IBM.Application:db2_db2inst3_2-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs
   |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node1
```

```

'- Online IBM.Application:db2mnt-database_db2inst3_NODE0002-rs:node2
Pending online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Offline IBM.Application:db2_db2inst3_3-rs
|  |- Offline IBM.Application:db2_db2inst3_3-rs:node1
|  '- Offline IBM.Application:db2_db2inst3_3-rs:node2
'- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs
|  |- Offline IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node1
|  '- Online IBM.Application:db2mnt-database_db2inst3_NODE0003-rs:node2

```

---

To recover from this state, just start the DB2 instance again:

```
(N1) $db2start
```

### ***Move DB2 partitions from one node to the other node***

If you need to stop DB2 on one of the partitioned database nodes for maintenance, you can use `db2haicu` to move the database partitions to the other nodes. The `db2haicu` command can be started from any physical database partition node in the cluster.

While moving partitions, you can access the data on the alive partition, but the moving partition cannot be accessed. Therefore, we recommend that you do *not* execute `db2haicu` to move the partitions during production time of the system.

Example 12-135 shows the sequences of moving partition2 and partition3 from node2 to node1 using `db2haicu`.

#### *Example 12-135 output of db2haicu to move partitions*

---

```
(N2) $db2haicu
```

```
...
```

Select an administrative task by number from the list below:

1. Add or remove cluster nodes.
2. Add or remove a network interface.
3. Add or remove DB2 database partitions.
4. Add or remove a highly available database.
5. Add or remove a mount point.
6. Add or remove an IP address.
7. Add or remove a non-critical path.
8. Move DB2 database partitions and HADR databases for scheduled maintenance.
9. Change failover policy for this instance.
10. Create a new quorum device for the domain.
11. Destroy the domain.
12. Exit.

Enter your selection:

```
8
```

Do you want to review the status of each cluster node in the domain before you begin?

```
[1]
```

```
1. Yes
```

```
2. No
```

```
1
```

```
Domain Name: dpf_domain
```



```
Node Name: node2 --- State: Online
Node Name: node1 --- State: Online
Enter the name of the node away from which DB2 partitions and HADR primary roles should
be moved:
```

### **node2**

Cluster node node2 hosts the following resources associated with the database manager instance db2inst3:

```
DB2 database partition number 3
```

```
DB2 database partition number 2
```

All of these cluster resource groups will be moved away from the cluster node node2. This will take their database partitions offline for the duration of the move, or cause an HADR role switch for HADR databases. Clients will experience an outage from this process. Are you sure you want to continue? [1]

1. Yes

2. No

**1**

```
Submitting move request for resource group db2_db2inst3_3-rg ...
```

```
The move request for resource group db2_db2inst3_3-rg was submitted successfully.
```

```
Submitting move request for resource group db2_db2inst3_2-rg ...
```

```
The move request for resource group db2_db2inst3_2-rg was submitted successfully.
```

Do you want to make any other changes to the cluster configuration? [1]

1. Yes

2. No

**2**

```
All cluster configurations have been completed successfully. db2haicu exiting ...
```

---

In this case, TSA executes the failover process:

- ▶ TSA moves the resource group db2\_db2inst3\_2-rg and db2\_db2inst3\_3-rg from node2 to node1.
- ▶ TSA invokes the start script on the new owner node (node1) to restart DB2 partition2 and partition3.

The scripts include the **db2gcf** command to restart the failed partitions (partition2 and partition3) and change the db2nodes.cfg file to restart the partitions. For example:

Before the takeover:

```
(N1) #cat db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node2 0 node2_int
3 node2 1 node2_int
```

After the takeover:

```
(N1) #cat db2nodes.cfg
0 node1 0 node1_int
1 node1 1 node1_int
2 node1 2 node1_int
3 node1 3 node1_int
```

After successfully moving the partitions, all the partitions can be accessed.

You can also use `db2haicu` to move the only two partitions back to the previous node as shown in Example 12-136. Note that in Example 12-135 on page 622, we move all partitions on one node (node2). However, in Example 12-136, we are only moving some partitions (only two of four) on one node (node1). So we select **No** for the choice. All of these cluster resource groups are moved away.

*Example 12-136 output of db2haicu to take back the tow partitions*

---

```
(B) db2haicu
...
Select an administrative task by number from the list below:
...
Enter your selection:
8
Do you want to review the status of each cluster node in the domain before you begin?
[1]
1. Yes
2. No

Domain Name: dpf_domain
  Node Name: Zaire --- State: Online
  Node Name: Baltic --- State: Online
Enter the name of the node away from which DB2 partitions and HADR primary roles should
be moved:
Baltic
Cluster node Baltic hosts the following resources associated with the database manager
instance db2inst3:
  DB2 database partition number 3
  DB2 database partition number 2
  DB2 database partition number 1
  DB2 database partition number 0
All of these cluster resource groups will be moved away from the cluster node Baltic.
This will take their database partitions offline for the duration of the move, or cause
an HADR role switch for HADR databases. Clients will experience an outage from this
process. Are you sure you want to continue? [1]
1. Yes
2. No
2
  DB2 database partition number 2
1. Yes
2. No
1
Submitting move request for resource group db2_db2inst3_2-rg ...
The move request for resource group db2_db2inst3_2-rg was submitted successfully.
  DB2 database partition number 3
1. Yes
2. No
1
Submitting move request for resource group db2_db2inst3_3-rg ...
The move request for resource group db2_db2inst3_3-rg was submitted successfully.
```

```

DB2 database partition number 1
1. Yes
2. No
2
Do you want to make any other changes to the cluster configuration? [1]
1. Yes
2. No
2
All cluster configurations have been completed successfully. db2haicu exiting ...

```

---

## Manual OS operations

In this section we describe how to stop and restart all nodes for maintenance at the operating system level.

To stop all the nodes, perform the following steps:

1. Stop the all resource groups.

Use the **chrg** command on either node to stop the resource groups. You need to stop the DB2 resource groups first, then stop the NFS server resource group (SA-nfssserver-rg).

For example, execute the **chrg** command on node1 to stop the DB2 resource groups. After that, all the DB2 partitions are stopped gracefully:

```

(N1) #chrg -o offline db2_db2inst3_3-rg
(N1) #chrg -o offline db2_db2inst3_2-rg
(N1) #chrg -o offline db2_db2inst3_1-rg
(N1) #chrg -o offline db2_db2inst3_0-rg

```

Confirm that all the nodes are stopped by using the **Issam** command.

Execute the **chrg** command on node1; stop the NFS server resource group:

```

(N1) #chrg -o offline SA-nfssserver-rg

```

2. Stop the cluster domain.

Issue the **stoprpdomain** command to stop the cluster domain (dpf\_domain) as the root user on either node:

```

stoprpdomain domainname
(N1) #stoprpdomain dpf_domain
(N1) #lsrpdomain
Name           OpState  RSCTActiveVersion  MixedVersions  TSPort  GSPort
dpf_domain  Offline  2.4.8.3             No              12347  12348

```

To restart the nodes, use these steps:

1. Start the cluster domain.

Issue the **starttrpdomain** command on either node.

```
starttrpdomain domainname
(N1) #starttrpdomain dpf_domain
(N1) !srpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
dpf_domain Online  2.4.8.3           No           12347  12348
```

2. Start the resource groups:

Issue the **chrg** command on either node to start the resource groups. In the restart process, you should to start the NFS server resource group (SA-nfssserver-rg) first then start the DB2 resource groups, because we need instance home directory to start DB2.

For example, to start the NFS server resource group:

```
(N1) #chrg -o online SA-nfssserver-rg
```

To confirm that the resource group is started successfully, use the **!ssam** command to check the status.

Confirm that /home/db2inst3 directory is mounted by the **df** command on each node. If the directory is not automatically mounted, you execute the **mount** command on each node.

```
(N1)(N2) #mount /home/db2inst3
```

Start the DB2 resource groups on either node. We recommend that you start partition0 first because partition0 is a catalog partition:

```
(N1) #chrg -o online db2_db2inst3_0-rg
(N1) #chrg -o online db2_db2inst3_1-rg
(N1) #chrg -o online db2_db2inst3_2-rg
(N1) #chrg -o online db2_db2inst3_3-rg
```

Again, use the **!ssam** command to observe the state of the resources.

## 12.7 DB2 partitioned database HA configuration with shared disk on Linux

This section describes how to configure an automated cluster controlled DB2 multi-partitioned database using the **db2haicu** utility in a Linux environment.

In a multi-partitioned database with TSA environment, TSA facilitates the detection of failures and automatically perform resource failover from one node to another within the cluster.

## 12.7.1 Architecture

Before you start the setup, you should plan the cluster environment. The overall planning considerations for the Linux environment are the same as those for the AIX platform. We discuss this briefly in 12.6.1, “Architecture” on page 581.

### Lab environment

Figure 12-14 illustrates the configuration of the multi-partitioned database with TSA in our lab environment.

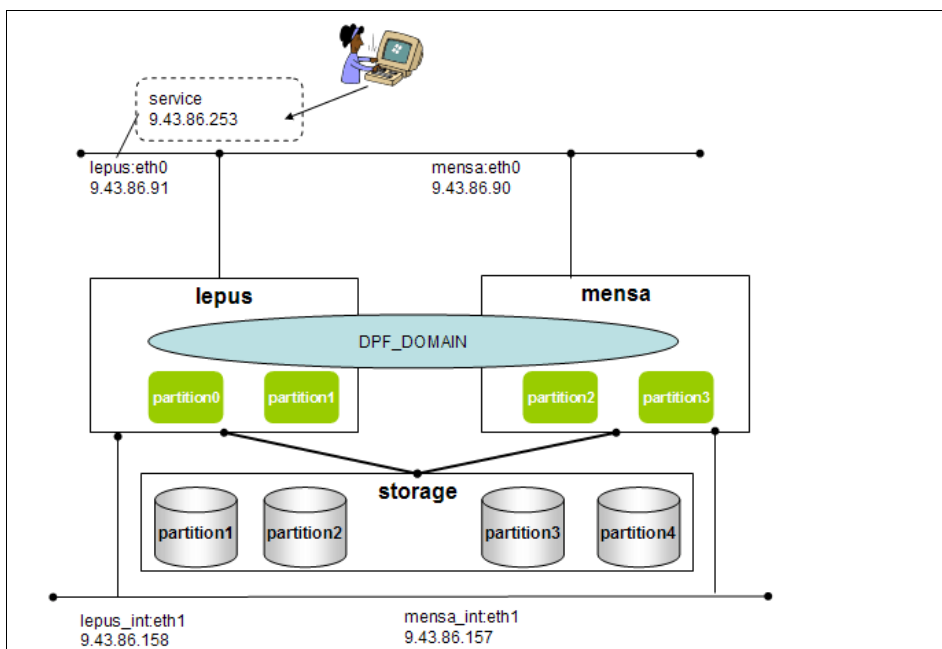


Figure 12-14 Multi-partitioned database with TSA Lab setup

The planning for our lab environment is as follows:

- ▶ Physical nodes configuration:
  - Two physical nodes named **lepus** and **mensa** are defined in the cluster.
  - The software used to set up the lab environment are:
    - SLES10 U1 (2.6.16.46-0.12-smp #1)
    - DB2 9.5 Enterprise Edition 9.5 Fixpack 1
    - Tivoli System Automation for Multiplatforms (SA MP) Base Component V2.2 FP6
    - RSCT 2.5.1.1

► Network configuration:

The nodes are connected to each other using two distinct networks:

- Public network: Defined to host the virtual IP address that allows clients to connect to the catalog partition. Here is the setup for our lab sample:

```
Primary node (lepus)
eth0: 9.43.86.91 (255.255.252.0)
Standby node (mensa)
eth0: 9.43.86.90 (255.255.252.0)
```

- Private network: Defined to carry out inter-partition communication (FCM). Here is the setup for our lab sample:

```
Primary node (lepus)
eth1: 192.168.86.91 (255.255.252.0)
Standby node (mensa)
eth1: 192.168.86.90 (255.255.252.0)
```

► Database configuration:

- Each node has a DB2 instance named *db2inst3*.
  - lepus has two logical partitions. Partition0 is a catalog partition that has system catalog tables and some user data. Partition1 is a data partition to store the user data.
  - mensa has two logical partitions, partition2 and partition3. Both are data partitions containing user data.
- A database named *SAMPLE* is created across these partitions.

## 12.7.2 Configuration

In this section, we discuss how to configure the Linux system and DB2 partitioned database environment. We summarize the procedures as follows:

► Linux system setup and configuration:

- a. Configure the Network Time Protocol (NTP) server to synchronize time across the cluster.
- b. Set up GPFS™.
- c. Tune the Linux kernel parameters.
- d. Configure volume groups and file systems
- e. Create the required groups and users.
- f. Install DB2 and TSA software on all nodes.

► DB2 DPF setup and configuration:

- a. Create the DB2 instance and configure it for communications.
- b. Set up remote shell - rsh

- c. Configure the `/etc/services` files for all servers in the cluster.
- d. Set up the `/etc/hosts` file.
- e. Define the database partition servers in the `db2nodes.cfg` file.
- f. Start the instance and verify that all database partitions are started.
- g. Create the database.
- h. Set up the automatic client reroute (optional)

## Linux system setup and configuration

This section provides an overview of the Linux setup and configuration tasks for the DB2 multi-partitioned database environment.

### **Network configuration**

The first task is to configure the Network Time Protocol (NTP) server to synchronize time across the cluster. You should maintain synchronized system clocks across the database partition servers to ensure smooth database operations and unlimited forward recoverability. Time differences among the database partition servers, plus any potential operational and communications delays for a transaction, should be less than the value specified for the `MAX_TIME_DIFF` (maximum time difference among nodes) database manager configuration parameter. The default value is 60 minutes.

To ensure that all the system clocks in the cluster are synchronized, you should set up Network Time Protocol (NTP) by setting up a server as an NTP server and synchronize all nodes to this server.

Perform the following tasks on an NTP server. We designated *lead* as an NTP server in our lab:

1. On all cluster nodes and NTP server, back up the existing NTP configuration file:

```
cp /etc/ntp.conf /etc/ntp.conf.ORIG
```

2. Check the NTP daemon:

```
chkconfig --list ntp
```

The output should look similar to this:

```
ntp 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

3. On the `/etc/ntp.conf` file, update the following contents:

```
server 127.127.1.0          # local clock (LCL)
fudge 127.127.1.0 stratum 10 # LCL is unsynchronized
driftfile /var/lib/ntp/drift/ntp.drift
logfile   /var/log/ntp
```

Ensure that the drift file exists on all nodes. If it does not, create it:

```
touch /var/lib/ntp/drift/ntp.drift
```

4. Enable the NTP daemon during system startup on all the nodes:

```
chkconfig --level 345 ntp on
```

5. Check the NTP daemon:

```
chkconfig --list ntp
```

The output should look similar to this:

```
ntp 0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

6. Start the NTP daemon on the NTP server:

```
rcntp start
```

The output should look similar to this:

```
Starting network time protocol daemon (NTPD) done
```

7. Check the status of the NTP daemon:

```
rcntp status
```

The output should look similar to this:

```
Checking for network time protocol daemon (NTPD): running
```

8. Set the time and date on the NTP server using the date command:

```
date MMDDhhmm
```

9. Check the system date and time:

```
date
```

10. Write the system date and time to the system clock using the following command on the NTP server:

```
hwclock -w
```

11. Test the localhost as follows:

```
ntpdate -q localhost
```

If NTP is working, the output looks similar to this:

```
server 127.0.0.1, stratum 11, offset 0.000000, delay 0.02563
server ::1, stratum 11, offset 0.000002, delay 0.02563
 5 May 22:16:34 ntpdate[2761]: adjust time server ::1 offset
0.000002 sec
```

If NTP is not working, you see the following message:

```
no server suitable for synchronization found
```



Now, on all cluster nodes, perform these tasks:

12. Update the value of the entry *server* to the IP address of *lead* on the `/etc/ntp.conf` file as follows:

```
server 9.43.86.68
driftfile /var/lib/ntp/drift/ntp.drift
logfile /var/log/ntp.log
```

13. Check the NTP daemon by running the following command:

```
chkconfig --list ntp
```

You should see output similar to this:

```
ntp                0:off 1:off 2:off 3:on  4:off 5:on  6:off
```

14. Start the NTP daemon by running:

```
rcntp start
```

15. Check the status of the NTP daemon:

```
rcntp status
```

16. Check the system date and time on all nodes using the **date** command.

17. Write the system date and time to the system clock using the following command:

```
hwclock -w
```

18. Query the NTP server from all nodes as follows:

```
ntpdate -q lead
```

The output looks similar to this:

```
server 9.43.86.68, stratum 11, offset -0.001754, delay 0.02577
 5 May 22:29:37 ntpdate[24256]: adjust time server 9.43.86.68 offset
-0.001754 sec
```

If NTP is not working, you receive the following output:

```
no server suitable for synchronization found
```

### **Set up GPFS**

Note that we are using GPFS for home directory share purposes. You could use NFS in a way similar to what is documented for AIX above; however for this Linux example we chose to use GPFS.

To set up IBM General Parallel File System™ (GPFS) for DB2 instance home directory, refer the GPFS V3.2 documentation at:

<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfsbooks.html>

## Tune the Linux kernel parameters

To tune the applicable kernel parameters, refer to “Chapter 6. Implementation part 2: Linux system setup and configuration” of *IBM Balanced Warehouse D5100: Design and Implementation, SC23-7735-00*.

## Configure volume groups and file systems

For more information, refer to 12.5.2, “Configuration” on page 560 to configure volume group, file systems, and multi-path devices. Example 12-137 here shows a sample output of multi-path devices in our environment.

### Example 12-137 Sample output of /dev/disk/by-name

---

```
mensa:/dev/disk/by-name # ll
lrwxrwxrwx 1 root root 11 Jun 8 2011 datavg-db2audit -> ../../dm-21
lrwxrwxrwx 1 root root 11 Jun 8 2011 datavg-db2data1v1 -> ../../dm-19
lrwxrwxrwx 1 root root 11 Jun 8 2011 datavg-db2dump -> ../../dm-22
lrwxrwxrwx 1 root root 11 Jun 8 2011 datavg-db2log1v1 -> ../../dm-20
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2homevg-db2home1v -> ../../dm-18
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp0-db2datap0 -> ../../dm-16
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp0-db2logp0 -> ../../dm-17
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp1-db2datap1 -> ../../dm-14
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp1-db2logp1 -> ../../dm-15
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp2-db2datap2 -> ../../dm-12
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp2-db2logp2 -> ../../dm-13
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp3-db2datap3 -> ../../dm-10
lrwxrwxrwx 1 root root 11 Jun 8 2011 db2vgp3-db2logp3 -> ../../dm-11
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath1 -> ../../dm-0
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath10 -> ../../dm-9
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath2 -> ../../dm-1
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath3 -> ../../dm-2
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath4 -> ../../dm-3
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath5 -> ../../dm-4
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath6 -> ../../dm-5
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath7 -> ../../dm-6
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath8 -> ../../dm-7
lrwxrwxrwx 1 root root 10 Jun 8 2011 mpath9 -> ../../dm-8
```

---

You can check information about multipathed devices through the `multipath -ll` command. Example 12-138 shows a sample of the output in our environment.

### Example 12-138 Information of multipathed devices

---

```
mensa:/dev/disk/by-name # multipath -ll
mpath2 (360050768018600c4700000000000091) dm-1 IBM,2145
[size=150G][features=0][hwhandler=0]
  _ round-robin 0 [prio=100][active]
    _ 4:0:0:0 sdc 8:32 [active][ready]
    _ 4:0:2:0 sde 8:64 [active][ready]
  _ round-robin 0 [prio=20][enabled]
    _ 4:0:1:0 sdd 8:48 [active][ready]
    _ 4:0:3:0 sdf 8:80 [active][ready]
mpath1 (SATA_WDC_WD5000AAKS-_WD-WCAPW0990173) dm-0 ATA,WDC WD5000AAKS-6
[size=466G][features=0][hwhandler=0]
  _ round-robin 0 [prio=0][active]
    _ 2:0:0:0 sdb 8:16 [active][ready]
mpath9 (3600a0b800026b2820000307f47f4d5a7) dm-8 IBM,Virtua1Disk
[size=29G][features=0][hwhandler=0]
  _ round-robin 0 [prio=0][active]
```

Table 12-6 lists the storage resources we created for this scenario.

Table 12-6 Storage resources

Multipathed devices	Volume groups	Logical volumes	File systems
/dev/mapper/ db2vgp0-db2datap0	db2vgp0	/dev/db2vgp0/db2datap0	/db2data/db2inst3/ NODE0000
/dev/mapper/ db2vgp1-db2datap1	db2vgp1	/dev/db2vgp1/db2datap1	/db2data/db2inst3/ NODE0001
/dev/mapper/ db2vgp2-db2datap2	db2vgp2	/dev/db2vgp2/db2datap2	/db2data/db2inst3/ NODE0002
/dev/mapper/ db2vgp3-db2datap3	db2vgp3	/dev/db2vgp3/db2datap3	/db2data/db2inst3/ NODE0003

Mount the file systems manually through the **mount** command. Example 12-139 shows the entries in the `/etc/fstab` file. Note that you should use the **noauto** mount option, disable the file systems dump, and **fsck** by specifying 0 for the fifth and sixth fields.

Example 12-139 Mount points in `/etc/fstab`

```
/dev/db2vgp0/db2datap0 /db2data/db2inst3/NODE0000 ext3 acl,user_xattr,noauto 0 0
/dev/db2vgp1/db2datap1 /db2data/db2inst3/NODE0001 ext3 acl,user_xattr,noauto 0 0
/dev/db2vgp2/db2datap2 /db2data/db2inst3/NODE0002 ext3 acl,user_xattr,noauto 0 0
/dev/db2vgp3/db2datap3 /db2data/db2inst3/NODE0003 ext3 acl,user_xattr,noauto 0 0
```

Activate volume groups and mount the file systems on the appropriate node as shown in Example 12-140.

Example 12-140 Activate VGs and mount file systems

```
lepus:/ # vgchange -an db2vgp0
lepus:/ # vgchange -an db2vgp1
lepus:/ # vgchange -an db2vgp2
lepus:/ # vgchange -an db2vgp3
lepus:/ # mount /db2data/db2inst3/NODE0000
lepus:/ # mount /db2data/db2inst3/NODE0001
mensa:/ # mount /db2data/db2inst3/NODE0002
mensa:/ # mount /db2data/db2inst3/NODE0003
```

## **Create the required operating system groups and users**

Perform the following steps:

1. Create the three required user groups:
  - db2iadm1 for DB2 instance user
  - db2fadm1 for DB2 fenced user
  - dasadm1 for DB2 administration server user

These user groups should exist on all database partition nodes in the cluster. It is essential that a group have the same group ID across all nodes.

2. Create the three required users:
  - db2inst3 as DB2 instance owner
  - db2fenc3 as the DB2 fenced user
  - db2das3 as DB2 administration server user.

These users should exist on all database partition nodes in the cluster. It is essential that a user have the same user ID across all nodes.

3. To set the password for all users, use the **passwd** command.

## **Install DB2**

We install Enterprise Edition 9.5 Fixpack 1 on all nodes. For the details of installing and setting up a multi-partitioned database on Linux, refer to *Up and Running with DB2 for Linux*, SG24-6899. We install DB2 in this step only and discuss the instance creation and configuration in the next section.

## **DB2 multi-partitioned database setup and configuration**

In this section, we discuss how to configure a multi-partitioned database and the remote communication.

1. Create the DB2 instance and configure it for communications.
  - a. On the machine which you want to be an instance owning machine, create the instance using the db2icrt command:

```
# /opt/IBM/db2/V9.5/instance/db2icrt -u db2fenc3 db2inst3
```

Node "lepus" is lab's instance owning machine.

- b. The partitioned database environment uses TCP/IP communications for remote client connections, and for sending data between partitions when needed. You should ensure that TCP/IP is enabled for all networks on all cluster nodes using the command:

```
# db2set DB2COMM=tcpip
```

- c. You update the SVCENAME database manager configuration parameter using the appropriate value listed in the /etc/services file. In this section, where the instance name is db2inst3 and the service name is db2c\_db2inst3, the command would look like this:

```
#db2 update dbm config using svcename db2c_db2inst3
```

- d. You disable the fault monitor at the instance level using the command:

```
# db2fm -i db2inst3 -f no
```

2. Set up remote shell - rsh.

In a partitioned DB2 environment, each database partition must have the authority to execute remote commands on all the participating machines without a password. We must add the DB2 instance owner and host name as trusted remote users across all participating machines. To do this, update the .rhosts file of the instance owner. The format is:

```
hostname instance-owner-name
```

Because the DB2 instance home directory is a GPFS shared directory, we only need to edit it once. In our case, we added the following entries to the .rhosts file as shown in Example 12-141.

*Example 12-141 .rhosts entries*

---

```
mensa.itsosj.sanjose.ibm.com    db2inst3
mensa_int.itsosj.sanjose.ibm.com db2inst3
lepus.itsosj.sanjose.ibm.com    db2inst3
lepus_int.itsosj.sanjose.ibm.com db2inst3
```

---

You need to change the .rhosts file permission to 600:

```
chmod 600 ~/db2inst1/.rhosts
```

Now test whether you can execute the command through rsh without a password among participating machines:

```
$ rsh mensa date
Thu May  8 18:03:30 PDT 2008
```

3. Configure the /etc/services files for all servers in the cluster.

You enable communication between database partition servers that participate in your partitioned database system. Communication between database partition servers is handled by the Fast Communications Manager (FCM). To enable FCM, a port or port range must be reserved in the /etc/services file on each computer in your partitioned database system. The Fast Communication Manager uses these ports. The reserved ports are in the following format (port numbers can vary):

```
DB2_cdb2inst3    50003/tcp
DB2_db2inst3     60012/tcp
```

```
DB2_db2inst3_1 60013/tcp
DB2_db2inst3_2 60014/tcp
DB2_db2inst3_3 60015/tcp
DB2_db2inst3_END      60016/tcp
```

The services file must be updated with the correct number of entries depending on the number of partitions being configured for the DPF instance. When partitions within the same server are involved, communication between the partitions still requires this setup.

**Note:** If the environment includes a high availability (HA) solution that is configured to fail over database partitions from one server to another, the `/etc/services` file must contain one FCM entry for each database partition that the server could potentially host after a failover occurs.

#### 4. Set up the `/etc/hosts` file.

Ensure that matching host name entries exist in `/etc/hosts` files on all cluster nodes. Entries for our configuration are shown in Example 12-142.

*Example 12-142 /etc/hosts entries*

---

```
9.43.86.90      mensa.itsosj.sanjose.ibm.com
9.43.86.91      lepus.itsosj.sanjose.ibm.com lepus
9.43.86.157    mensa_int.itsosj.sanjose.ibm.com mensa_int
9.43.86.158    lepus_int.itsosj.sanjose.ibm.com lepus_int
```

---

#### 5. Define the database partition servers in the `db2nodes.cfg` file.

The mapping of database partitions to servers is specified in the `db2nodes.cfg` file, which is found in the instance home directory (that is, `~/sqllib/db2nodes.cfg`). The plan for database partition numbering is as follows:

Database partition0 has the following characteristics:

- Catalog function (only one database partition has the database catalog)
- Coordinator function

Database partition1 through partition3 have the following characteristics:

- Database partitions with partitioned data

When you create an instance, the `db2nodes.cfg` file is automatically created and an entry for the instance-owning database partition server is added. For example, when you create the DB2 instance, on `lepus`, the `db2nodes.cfg` file looks like this:

```
#cat db2nodes.cfg
0 lepus 0
```

This entry includes the database partition server number (node number), the TCP/IP host name of the server where the database partition server resides, a logical port number for the database server partition, and a netname for the host name or the IP address of the high speed interconnect for FCM communication, if you have the high speed interconnect.

If you are installing a partitioned database with a configuration the same as the one shown in Figure 12-10 on page 582, with four computers and a database partition server on each computer, the updated db2nodes.cfg should appear similar to the following:

```
cat db2nodes.cfg
0 lepus 0 lepus_int
1 lepus 1 lepus_int
2 mensa 0 mensa_int
3 mensa 1 mensa_int
```

6. Start the instance and verify that all database partitions are started.

7. Create the database spanning across the above four partitions.

```
db2samp1 -dbpath /db2data
```

8. Set up the client reroute for a minimum downtime for client connections (optional). We use a virtual IP address that allows clients to connect to the catalog node. Identify the port number for the instance TCP/IP listener by checking the value of the SVCENAME **dbm cfg** parameter and use it with the following command.

```
db2 update alternate server for database sample using hostname
9.43.86.253 port 50003
```

You can check the value of the alternate server host name and port number by using the following command:

```
db2inst3@mensa:~/sprabhu> db2 list db directory
System Database Directory
Number of entries in the directory = 1
```

Database 1 entry:

Database alias	= SAMPLE
Database name	= SAMPLE
Local database directory	= /db2data
Database release level	= c.00
Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	= 9.43.86.253
Alternate server port number	= 50003

In summary, Figure 12-15 shows the hosts and services entries on both nodes along with db2nodes.cfg.

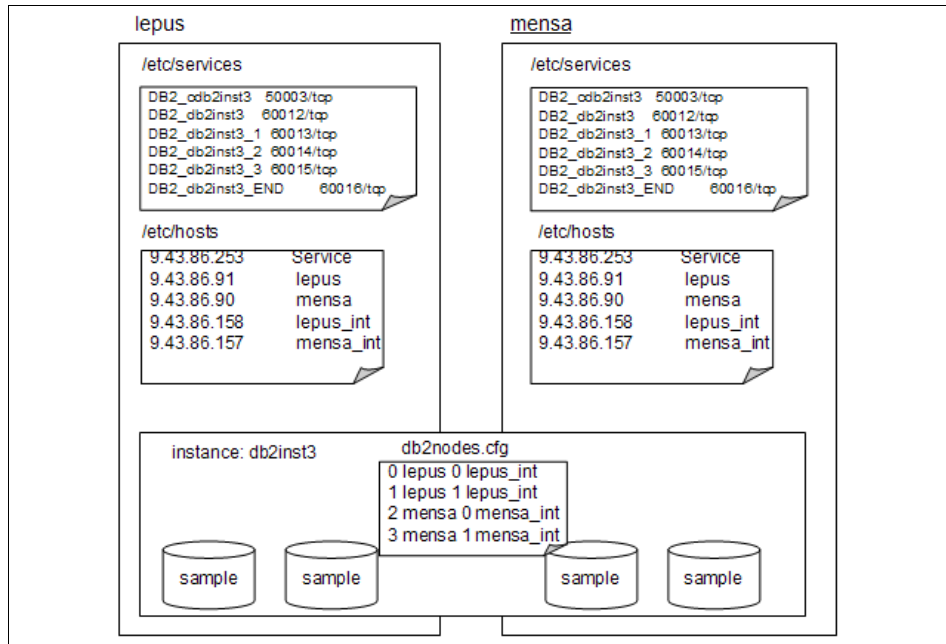


Figure 12-15 Partitioned database Configuration

When the DB2 multi-partitioned database setup is complete, it is ready to be made highly available, as discussed in the following section.

### 12.7.3 Cluster domain configuration using db2haicu

This section describes how to configure a partitioned instance for high availability. After the initial configuration, TSA automatically responds to the unexpected failure of a server by moving the database partition resources from the failed nodes to its failover pair.

**Note:** To prevent over-committing memory on the node to which the database partitions were failed over, you should reduce the buffer pools and sort heap threshold in order to accommodate the additional data partitions.

To prepare each node for clustering, run `preprnode` on all servers as user `root`:

```

/usr/sbin/rsct/bin/preprnode lepus mensa
  
```



## Creating a cluster domain using db2haicu interactive mode

After the preceding preliminary configuration steps are completed, we can enable high availability and automate the failover using the db2haicu utility. It can be run on any cluster node. The details involving the process are outlined here:

1. Start with the welcome screen.

Log on to any of the nodes and issue the **db2haicu** command from the command line prompt. **db2haicu** displays the welcome screen message as shown in Example 12-143.

### *Example 12-143 Welcome screen*

---

```
db2inst3@1epus:~> db2haicu
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

```
db2haicu determined the current DB2 database manager instance is db2inst3. The
cluster configuration that follows will apply to this instance.
```

```
db2haicu is collecting information on your current setup. This step may take some
time as db2haicu will need to activate all databases for the instance to discover all
paths ...
```

```
When you use db2haicu to configure your clustered environment, you create cluster
domains. For more information, see the topic 'Creating a cluster domain with
db2haicu' in the DB2 Information Center. db2haicu is searching the current machine
for an existing active cluster domain ...
```

```
db2haicu did not find a cluster domain on this machine. db2haicu will now query the
system for information about cluster nodes to create a new cluster domain ...
```

```
db2haicu did not find a cluster domain on this machine. To continue configuring your
clustered environment for high availability, you must create a cluster domain;
otherwise, db2haicu will exit.
```

```
Create a domain and continue? [1]
```

1. Yes
  2. No
- 

Note that the number contained within square brackets is the default for that prompt. Simply press Enter to select the default value.

To create a cluster domain, enter **1** at the initial prompt. **db2haicu** then prompts for the domain name and host name information. Provide all the input values. See Example 12-144.

### Example 12-144 Input domain name

---

Create a unique name for the new domain:

**DPF\_DOMAIN**

Nodes must now be added to the new domain.

How many cluster nodes will the domain DPF\_DOMAIN contain?

**2**

Enter the host name of a machine to add to the domain:

**lepus**

Enter the host name of a machine to add to the domain:

**Mensa**

db2haicu can now create a new domain containing the 2 machines that you specified. If you choose not to create a domain now, db2haicu will exit.

Create the domain now? [1]

1. Yes

2. No

**1**

Creating domain DPF\_DOMAIN in the cluster ...

Creating domain DPF\_DOMAIN in the cluster was successful.

---

## 2. Specify the quorum configuration.

After completing the domain creation, you must configure a quorum for the cluster domain. The supported quorum type for this solution is a *network quorum*. A network quorum (or network tiebreaker) is a pingable IP address that is used to decide which node in the cluster serves as the “active” node during a site failure, and which nodes are offline. Note that the machine hosting this IP address does not need any particular software or operating system level installed; its primary requirement is that it can be pinged from all nodes in the cluster, and must remain pingable in the case of cluster node failures.

You are prompted by **db2haicu** to enter quorum configuration values, such as the IP address for the network tiebreaker, as shown in Example 12-145.

### Example 12-145 Quorum configuration

---

You can now configure a quorum device for the domain. For more information, see the topic “Quorum devices” in the DB2 Information Center. If you do not configure a quorum device for the domain, then a human operator will have to manually intervene if subsets of machines in the cluster lose connectivity.

Configure a quorum device for the domain called DPF\_DOMAIN? [1]

1. Yes

2. No

The following is a list of supported quorum device types:

1. Network Quorum

Enter the number corresponding to the quorum device type to be used: [1]

Specify the network address of the quorum device:

### 9.43.85.1

Configuring quorum device for domain DPF\_DOMAIN ...

Configuring quorum device for domain DPF\_DOMAIN was successful.

---

### 3. Define the network setup.

After specifying the quorum configuration, you must define the public and the private networks of your system to **db2haicu**. If network failure detection is important to your configuration, you must follow the prompts and add the networks to the cluster at this point. All network interfaces are automatically discovered by the **db2haicu** tool. Example 12-146 shows the network setup prompts and our input.

#### *Example 12-146 Network setup*

---

The cluster manager found 4 network interface cards on the machines in the domain. You can use **db2haicu** to create networks for these network interface cards. For more information, see the topic 'Creating networks with **db2haicu**' in the DB2 Information Center.

Create networks for these network interface cards? [1]

1. Yes
2. No

Enter the name of the network for the network interface card: eth0 on cluster node: **mensa**

1. Create a new public network for this network interface card.
2. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node mensa to the network db2\_public\_network\_0? [1]

1. Yes
2. No

Adding network interface card eth0 on cluster node mensa to the network db2\_public\_network\_0 ...

Adding network interface card eth0 on cluster node mensa to the network db2\_public\_network\_0 was successful.

Enter the name of the network for the network interface card: eth0 on cluster node: **lepup**

1. db2\_public\_network\_0
2. Create a new public network for this network interface card.
3. Create a new private network for this network interface card.

Enter selection:

**1**

Are you sure you want to add the network interface card eth0 on cluster node lepup to the network db2\_public\_network\_0? [1]

1. Yes
2. No

```

Adding network interface card eth0 on cluster node lepus to the network
db2_public_network_0 ...
Adding network interface card eth0 on cluster node lepus to the network
db2_public_network_0 was successful.
Enter the name of the network for the network interface card: eth1 on cluster
node: lepus
1. db2_public_network_0
2. Create a new public network for this network interface card.
3. Create a new private network for this network interface card.
Enter selection:
3
Are you sure you want to add the network interface card eth1 on cluster node
lepus to the network db2_private_network_0? [1]
1. Yes
2. No

Adding network interface card eth1 on cluster node lepus to the network
db2_private_network_0 ...
Adding network interface card eth1 on cluster node lepus to the network
db2_private_network_0 was successful.
Enter the name of the network for the network interface card: eth1 on cluster
node: mensa
1. db2_private_network_0
2. db2_public_network_0
3. Create a new public network for this network interface card.
4. Create a new private network for this network interface card.
Enter selection:
1
Are you sure you want to add the network interface card eth1 on cluster node
mensa to the network db2_private_network_0? [1]
1. Yes
2. No

Adding network interface card eth1 on cluster node mensa to the network
db2_private_network_0 ...
Adding network interface card eth1 on cluster node mensa to the network
db2_private_network_0 was successful.

```

---

Note that it is not possible to add two NICs with different subnet masks and different assigned IP addresses to the same common network. For example, in this configuration, if you try to define eth0 and eth1 to the same network using **db2haicu**, the input is rejected.

#### 4. Specify the cluster manager selection.

After the network definitions, **db2haicu** prompts for the cluster manager software being used for the current HA setup. See Example 12-147.

### *Example 12-147 Select cluster manager*

---

```
Retrieving high availability configuration parameter for instance db2inst3 ...
The cluster manager name configuration parameter (high availability
configuration parameter) is not set. For more information, see the topic
"cluster_mgr - Cluster manager name configuration parameter" in the DB2
Information Center. Do you want to set the high availability configuration
parameter?
The following are valid settings for the high availability configuration
parameter:
  1.TSA
  2.Vendor
Enter a value for the high availability configuration parameter: [1]
1
Setting a high availability configuration parameter for instance db2inst3 to
TSA.
```

---

#### 5. Configure the failover policy.

Now you need to configure the failover policy for the instance db2inst3. The failover policy determines the machines on which the cluster manager restarts the database manager if the database manager goes offline unexpectedly. For our setup, we select option **3**. Note that the failover policy is a powerful concept for larger clusters (with more nodes and more partitions), but for a simple two-node DPF setup (such as this one), it is generally best to select option **3**. Example 12-148 shows the failover policy selection section.

### *Example 12-148 Failover policy*

---

```
The following are the available failover policies:
  1. Local Restart -- during failover, the database manager will restart in
place on the local machine
  2. Round Robin -- during failover, the database manager will restart on
any machine in the cluster domain
  3. Mutual Takeover -- during failover, the database partitions on one
machine will failover to a specific machine and vice versa (used with DPF
instances)
  4. M+N -- during failover, the database partitions on one machine will
failover to any other machine in the cluster domain (used with DPF
instances)
  5. Custom -- during failover, the database manager will restart on a
machine from a user-specified list
Enter your selection:
3
```

---

Then **db2haicu** prompts you to designate any noncritical mount points. For our setup, we chose to designate only one non-critical mount point. See Example 12-149.

*Example 12-149 Enter non-critical mount point*

---

You can identify mount points that are noncritical for failover. For more information, see the topic 'Identifying mount points that are noncritical for failover' in the DB2 Information Center. Are there any mount points that you want to designate as noncritical? [2]

1. Yes

2. No

**1**

Enter the full path of the mount to be made non-critical:

**/tmp**

---

6. Make database partitions highly available.

**db2haicu** then prompts you to make all the DB2 database partitions highly available. Answer yes and input the system pair information for each partition. It also prompts you to enter a virtual IP address for all partitions. Typically we recommend configuring a virtual IP address just for the catalog partition.

You must make sure that your IP address and subnet mask values are well formed and correspond with the subnet mask of the network you chose. All invalid inputs are rejected. Example 12-150 shows the prompts from **db2haicu** and our input.

*Example 12-150 Make database partitions highly available*

---

The following DB2 database partitions can be made highly available:

DB2 database partition number 0

DB2 database partition number 1

DB2 database partition number 2

DB2 database partition number 3

Do you want to make all these DB2 database partitions highly available? [1]

1. Yes

2. No

Mutual takeover policy was chosen. You must specify a system pair for partition 0.

Enter the host name of system 1:

**lepus**

Enter the host name of system 2:

**mensa**

Adding DB2 database partition 0 to the cluster ...

Adding DB2 database partition 0 to the cluster was successful.

Do you want to configure a virtual IP address for the DB2 partition: 0? [2]

1. Yes

```
2. No
1
Enter the virtual IP address:
9.43.86.253
Enter the subnet mask for the virtual IP address 9.43.86.253: [255.255.255.0]
255.255.252.0
Select the network for the virtual IP 9.43.86.253:
1. db2_private_network_0
2. db2_public_network_0
Enter selection:
2
Adding virtual IP address 9.43.86.253 to the domain ...
Adding virtual IP address 9.43.86.253 to the domain was successful.
Mutual takeover policy was chosen. You must specify a system pair for partition
1.
Enter the host name of system 1:
lepus
Enter the host name of system 2:
Mensa
Adding DB2 database partition 1 to the cluster ...
Adding DB2 database partition 1 to the cluster was successful.
Do you want to configure a virtual IP address for the DB2 partition: 1? [2]
1. Yes
2. No
2
Mutual takeover policy was chosen. You must specify a system pair for partition
2.
Enter the host name of system 1:
mensa
Enter the host name of system 2:
Lepus
Adding DB2 database partition 2 to the cluster ...
Adding DB2 database partition 2 to the cluster was successful.
Do you want to configure a virtual IP address for the DB2 partition: 2? [2]
1. Yes
2. No
2
Mutual takeover policy was chosen. You must specify a system pair for partition
3.
Enter the host name of system 1:
mensa
Enter the host name of system 2:
Lepus
Adding DB2 database partition 3 to the cluster ...
Adding DB2 database partition 3 to the cluster was successful.
Do you want to configure a virtual IP address for the DB2 partition: 3? [2]
1. Yes
2. No
```

## 7. Make databases highly available.

**db2haicu** then prompts you to make all active databases highly available. By answering this prompt, you basically add all the mount points for your highly available databases. In our setup, we had only one database **SAMPLE**. See Example 12-151.

### *Example 12-151 Make database highly available*

---

```
The following databases can be made highly available:
  Database: SAMPLe
Do you want to make all active databases highly available? [1]
1. Yes
2. No
1
Adding database SAMPLe to the cluster domain ...
Adding database SAMPLe to the cluster domain was successful.
All cluster configurations have been completed successfully. db2haicu exiting
...
```

---

With these, all cluster configurations are complete. Note that to display the state of the cluster, you can issue the TSA command **Issam** or the **db2pd -ha**. Example 12-152 shows the **Issam** command output to see the resources created during this process.

### *Example 12-152 Sample Issam output*

---

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_0-rs
    |- Online IBM.Application:db2_db2inst3_0-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
  |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
    |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
  '- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
    |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepup
    '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_1-rs
    |- Online IBM.Application:db2_db2inst3_1-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
    |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_2-rs
    |- Offline IBM.Application:db2_db2inst3_2-rs:lepup
    '- Online IBM.Application:db2_db2inst3_2-rs:mensa
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
    |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
    '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst3_3-rs
    |- Offline IBM.Application:db2_db2inst3_3-rs:lepup
    '- Online IBM.Application:db2_db2inst3_3-rs:mensa
```



```
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
  |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

The command **db2pd -ha** can also be issued from the instance owner ID to examine the state of the resources, Example 12-153 shows a sample output.

*Example 12-153 db2pd -ha sample output*

---

```
DB2 HA Status
Instance Information:
Instance Name           = db2inst3
Number Of Domains      = 1
Number Of RGs for instance = 4

Domain Information:
Domain Name            = DPF_DOMAIN
Cluster Version       = 2.5.1.1
Cluster State         = Online
Number of nodes       = 2

Node Information:
Node Name              State
-----
lepus                  Online
mensa                  Online

Resource Group Information:
Resource Group Name    = db2_db2inst3_3-rg
Resource Group LockState = Unlocked
Resource Group OpState = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 2
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  mensa
  lepus

Member Resource Information:
Resource Name          = db2mnt-db2data_db2inst3_NODE0003-rs
Resource State        = Online
Resource Type         = Mount
Mount Resource Path   = /db2data/db2inst3/NODE0003
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  mensa
  lepus

Resource Name          = db2_db2inst3_3-rs
Resource State        = Online
Resource Type         = DB2 Partition
DB2 Partition Number = 3
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  mensa
  lepus

Resource Group Name    = db2_db2inst3_2-rg
Resource Group LockState = Unlocked
Resource Group OpState = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 2
Number of Allowed Nodes = 2
```

```

Allowed Nodes
-----
mensa
lepup
Member Resource Information:
Resource Name           = db2mnt-db2data_db2inst3_NODE0002-rs
Resource State          = Online
Resource Type           = Mount
Mount Resource Path     = /db2data/db2inst3/NODE0002
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  mensa
  lepup

Resource Name           = db2_db2inst3_2-rs
Resource State          = Online
Resource Type           = DB2 Partition
DB2 Partition Number    = 2
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  mensa
  lepup

Resource Group Name     = db2_db2inst3_1-rg
Resource Group LockState = Unlocked
Resource Group OpState  = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 2
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepup
  mensa
Member Resource Information:
Resource Name           = db2mnt-db2data_db2inst3_NODE0001-rs
Resource State          = Online
Resource Type           = Mount
Mount Resource Path     = /db2data/db2inst3/NODE0001
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepup
  mensa

Resource Name           = db2_db2inst3_1-rs
Resource State          = Online
Resource Type           = DB2 Partition
DB2 Partition Number    = 1
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepup
  mensa

Resource Group Name     = db2_db2inst3_0-rg
Resource Group LockState = Unlocked
Resource Group OpState  = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 3
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepup
  mensa
Member Resource Information:

```

```

Resource Name           = db2mnt-db2data_db2inst3_NODE0000-rs
Resource State         = Online
Resource Type          = Mount
Mount Resource Path    = /db2data/db2inst3/NODE0000
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepus
  mensa

Resource Name           = db2_db2inst3_0-rs
Resource State         = Online
Resource Type          = DB2 Partition
DB2 Partition Number   = 0
Number of Allowed Nodes = 2
  Allowed Nodes
  -----
  lepus
  mensa

Resource Name           = db2ip_9_43_86_253-rs
Resource State         = Online
Resource Type          = IP

```

```

Network Information:
Network Name           Number of Adapters
-----
db2_private_network_0 2
  Node Name           Adapter Name
  -----
  lepus               eth1
  mensa               eth1

```

```

Network Name           Number of Adapters
-----
db2_public_network_0  2
  Node Name           Adapter Name
  -----
  mensa               eth0
  lepus               eth0

```

```

Quorum Information:
Quorum Name           Quorum State
-----
db2_Quorum_Network_9_43_85_1:8_46_46  Online
Fail                  Offline
Operator              Offline

```

Note that the foregoing cluster can also be created using an XML file. The XML file that we used in our lab is shown in Example 12-154. It contains all the information that **db2haicu** needs to know in order to make an instance highly available with a pre-existing database **SAMPLE**.

*Example 12-154 Sample XML file for cluster domain creation*

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
*****
-->
<!-- ===== -->
<!-- = DB2Cluster element = -->
<!-- = This element encapsulates the cluster configuration = -->

```

```

<!-- = specification for the instance = -->
<!-- ===== -->
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="db2ha.xsd" clusterManagerName="TSA" version="1.0">
  <!-- ===== -->
  <!-- = ClusterDomain element = -->
  <!-- = This element encapsulates the cluster configuration = -->
  <!-- = specification = -->
  <!-- = Creating cluster domain of name db2HADomain = -->
  <!-- = Creating an IP quorum device (IP 9.43.85.1) = -->
  <!-- = The IP must be pingable at all times by each of the nodes in = -->
  <!-- = the cluster domain = -->
  <!-- ===== -->
  <ClusterDomain domainName="DPF_DOMAIN">
    <Quorum quorumDeviceProtocol="network" quorumDeviceName="9.43.85.1"/>
    <!-- ===== -->
    <!-- = Physical network element = -->
    <!-- = The physical network specifies the network type, protocol = -->
    <!-- = IP address, subnet mask, and NIC name = -->
    <!-- = Create network of name db2_public_network_0 = -->
    <!-- ===== -->
    <PhysicalNetwork physicalNetworkName="db2_public_network_0" physicalNetworkProtocol="ip">
      <Interface interfaceName="eth0" clusterNodeName="mensa">
        <IPAddress baseAddress="9.43.86.90" subnetMask="255.255.252.0"
networkName="db2_public_network_0" />
      </Interface>
      <Interface interfaceName="eth0" clusterNodeName="lepus">
        <IPAddress baseAddress="9.43.86.91" subnetMask="255.255.252.0"
networkName="db2_public_network_0" />
      </Interface>
    </PhysicalNetwork>

    <PhysicalNetwork physicalNetworkName="db2_private_network_0" physicalNetworkProtocol="ip">
      <Interface interfaceName="eth1" clusterNodeName="mensa">
        <IPAddress baseAddress="192.168.86.90" subnetMask="255.255.252.0"
networkName="db2_private_network_0" />
      </Interface>
      <Interface interfaceName="eth1" clusterNodeName="lepus">
        <IPAddress baseAddress="192.168.86.91" subnetMask="255.255.252.0"
networkName="db2_private_network_0" />
      </Interface>
    </PhysicalNetwork>
  <!-- ===== -->
  <!-- = ClusterNodeName element = -->
  <!-- = The set of nodes in the cluster domain = -->
  <!-- = Here the defined set of nodes in the domain is = -->
  <!-- = lepus, mensa = -->
  <!-- ===== -->
  <ClusterNode clusterNodeName="lepus"/>
  <ClusterNode clusterNodeName="mensa"/>
</ClusterDomain>
<!-- ===== -->
<!-- = Failover policy element = -->
<!-- = The failover policy specifies the failover order of the = -->
<!-- = cluster nodes = -->
<!-- = In the current sample the failover policy is mutual take over = -->
<!-- ===== -->
<FailoverPolicy>
  <Mutual/>
</FailoverPolicy>
<!-- ===== -->
<!-- = DB2 Partition element = -->
<!-- = The DB2 partition type specifies a DB2 Instance Name, = -->
<!-- = partition number, virtual IP address, Subnet mask, = -->
<!-- = mount resource and the = -->
<!-- = mutual failover nodes (lepus, mensa) = -->
<!-- ===== -->

```

```

<DB2PartitionSet>
  <DB2Partition dbpartitionnum="0" instanceName="db2inst3">
    <!-- ===== -->
    <!-- = Service IP associated with the DB2 Partition = -->
    <!-- = 9.43.85.1 = -->
    <!-- ===== -->
    <!-- ===== -->
    <VirtualIPAddress baseAddress="9.43.86.253" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    <!-- ===== -->
    <!-- = Mounts used by the DB2 Partition 0 = -->
    <!-- ===== -->
    <Mount filesystemPath="/db2data/db2inst3/NODE0000"/>
    <MutualPair systemPairNode1="lepus" systemPairNode2="mensa"/>
  </DB2Partition>
  <DB2Partition dbpartitionnum="1" instanceName="db2inst3">
    <!-- ===== -->
    <!-- = Mounts used by the DB2 Partition 1 = -->
    <!-- ===== -->
    <Mount filesystemPath="/db2data/db2inst3/NODE0001"/>
    <MutualPair systemPairNode1="lepus" systemPairNode2="mensa"/>
  </DB2Partition>
  <DB2Partition dbpartitionnum="2" instanceName="db2inst3">
    <!-- ===== -->
    <!-- = Mounts used by the DB2 Partition 2 = -->
    <!-- ===== -->
    <Mount filesystemPath="/db2data/db2inst3/NODE0002"/>
    <MutualPair systemPairNode1="mensa" systemPairNode2="lepus"/>
  </DB2Partition>
  <DB2Partition dbpartitionnum="3" instanceName="db2inst3">
    <!-- ===== -->
    <!-- = Mounts used by the DB2 Partition 3 = -->
    <!-- ===== -->
    <Mount filesystemPath="/db2data/db2inst3/NODE0003"/>
    <MutualPair systemPairNode1="mensa" systemPairNode2="lepus"/>
  </DB2Partition>
</DB2PartitionSet>
<!-- ===== -->
<!-- = List of Databases to be configured for High Availability = -->
<!-- ===== -->
<HADBSet instanceName="db2inst3">
  <HADB databaseName = "SAMPLE" />
</HADBSet>
</DB2Cluster>

```

The existing values in the preceding file can be replaced to reflect your own configuration and environment. Next we describe what the different elements shown in the preceding XML file represent:

- The <ClusterDomain> element covers all cluster-wide information. This includes quorum information, cluster node information, and cluster domain name.
- The <PhysicalNetwork> sub-element of the ClusterDomain element includes all network information. This includes the name of the network and the network interface cards contained in it. We define our single public network using this element.
- The <FailoverPolicy> element specifies the failover order of the cluster nodes. Mutual is an Active/Passive pair.

- The <DB2PartitionSet> element covers the DB2 instance information. This includes the current DB2 instance name, the DB2 partition number, and the virtual IP address associated with the instance.
- The <HADBSet> element specifies the database name that is to be made highly available. It includes the current DB2 instance name.

To configure the system with db2haicu XML mode:

- a. Log on to the instance owning machine as an instance owner.
- b. Issue the following command:

```
db2haicu -f path to XML file
```

If an invalid input is encountered during the process, **db2haicu** exits with a non-zero error code. After the XML file has been processed, all mount points defined to the DB2 database manager are defined to the cluster manager.

Example 12-155 shows a sample output from running **db2haicu** in XML mode.

*Example 12-155 Sample output from running db2haicu in XML mode*

---

```
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

```
You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.
```

```
For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.
```

```
db2haicu determined the current DB2 database manager instance is db2inst3. The cluster configuration that follows will apply to this instance.
```

```
db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...
Creating domain DPF_DOMAIN in the cluster ...
Creating domain DPF_DOMAIN in the cluster was successful.
Configuring quorum device for domain DPF_DOMAIN ...
Configuring quorum device for domain DPF_DOMAIN was successful.
Adding network interface card eth0 on cluster node mensa to the network db2_public_network_0 ...
Adding network interface card eth0 on cluster node mensa to the network db2_public_network_0 was successful.
Adding network interface card eth0 on cluster node lepus to the network db2_public_network_0 ...
Adding network interface card eth0 on cluster node lepus to the network db2_public_network_0 was successful.
Adding network interface card eth1 on cluster node mensa to the network db2_private_network_0 ...
Adding network interface card eth1 on cluster node mensa to the network db2_private_network_0 was successful.
Adding network interface card eth1 on cluster node lepus to the network db2_private_network_0 ...
Adding network interface card eth1 on cluster node lepus to the network db2_private_network_0 was successful.
Adding DB2 database partition 0 to the cluster ...
Adding DB2 database partition 0 to the cluster was successful.
Adding DB2 database partition 1 to the cluster ...
Adding DB2 database partition 1 to the cluster was successful.
Adding DB2 database partition 2 to the cluster ...
Adding DB2 database partition 2 to the cluster was successful.
Adding DB2 database partition 3 to the cluster ...
Adding DB2 database partition 3 to the cluster was successful.
Adding database SAMPLE to the cluster domain ...
```

Adding database SAMPLE to the cluster domain was successful.  
All cluster configurations have been completed successfully. db2haicu exiting ...

---

We now proceed with testing of this configuration.

## 12.7.4 Unplanned operations

When the partitioned database instance is made highly available, we can test our automated HA environment for some unplanned outages. Issue the **lssam** command, and observe the output displayed on the screen. You see output similar to Example 12-156.

### *Example 12-156 lssam output after cluster creation*

---

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
   |- Online IBM.Application:db2_db2inst3_0-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepup
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
   |- Online IBM.Application:db2_db2inst3_1-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
   |- Offline IBM.Application:db2_db2inst3_2-rs:lepup
   '- Online IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
   |- Offline IBM.Application:db2_db2inst3_3-rs:lepup
   '- Online IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

Next we provide a brief description of the resources listed in the preceding example. These resources are under DB2 instance resource groups:

- ▶ `db2_db2inst3_0-rg`: This is the resource group for partition0 containing the following member resources:
  - `db2_db2inst3_0-rs`: Partition 0 resource
  - `db2mnt-db2data_db2inst3_NODE0000-rs`: Mount point for partition0
  - `db2ip_9_43_86_253-rs`: virtual IP address
- ▶ `db2_db2inst3_1-rg`: This is the resource group for partition1 containing the following member resources:
  - `db2_db2inst3_1-rs`: Partition1 resource
  - `db2mnt-db2data_db2inst3_NODE0001-rs`: Mount point for partition1
- ▶ `db2_db2inst3_2-rg`: This the resource group for partition2 containing the following member resources:
  - `db2_db2inst3_2-rs`: Partition2 resource
  - `db2mnt-db2data_db2inst3_NODE0002-rs`: Mount point for partition2
- ▶ `db2_db2inst3_3-rg`: This the resource group for partition3 containing the following member resources.
  - `db2_db2inst3_3-rs`: Partition3 resource
  - `db2mnt-db2data_db2inst3_NODE0003-rs`: Mount point for partition3

The resources created by `db2haicu` during the configuration can be in one of the following states:

- ▶ **Online**: The resource has been started and is functioning normally.
- ▶ **Offline**: The resource has been successfully stopped.
- ▶ **Failed Offline**: The resource has malfunctioned.

For an exhaustive explanation of the possible resource states, consult Chapter 3, *OpState Attribute* in:

[http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8272-02/en\\_US/PDF/HALBAU02.pdf](http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8272-02/en_US/PDF/HALBAU02.pdf)



The relationship between the networks and the resource groups is illustrated in Figure 12-16.

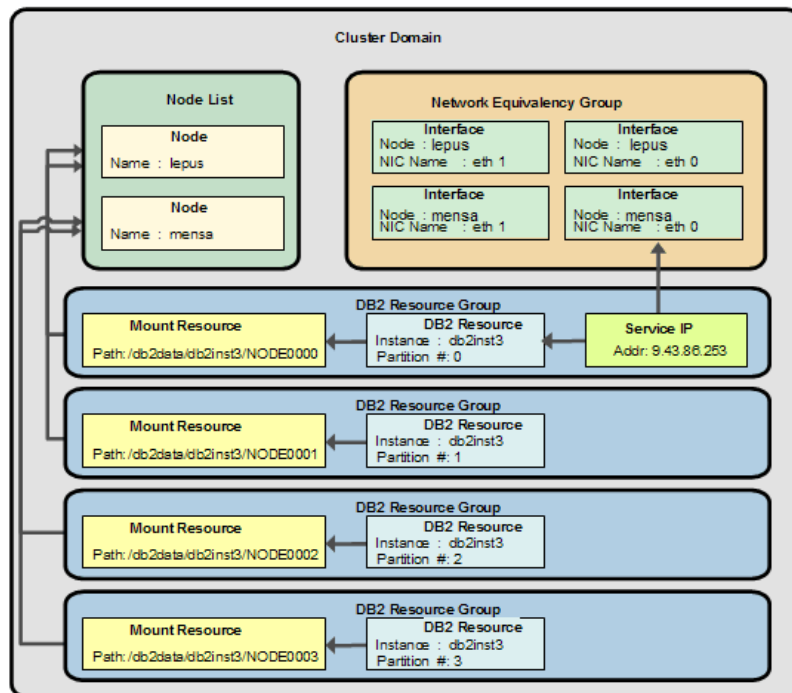


Figure 12-16 Resource relationship

In the following sections we explain how to deal with unplanned outages.

## Node failure

We discuss node failure in this section by simulating a node crash using either the power off method or execute the **shutdown** command on the node.

### **Non-catalog node failure**

We turn the power off on our non-catalog node mensa. The resources for partition2 and partition3 fail over to the pair node, lepus, and remain “online” there. The states of the resources on mensa are in a “Failed Offline” state until the power is restored.

Figure 12-17 shows the cluster behavior in the event of a node crash.

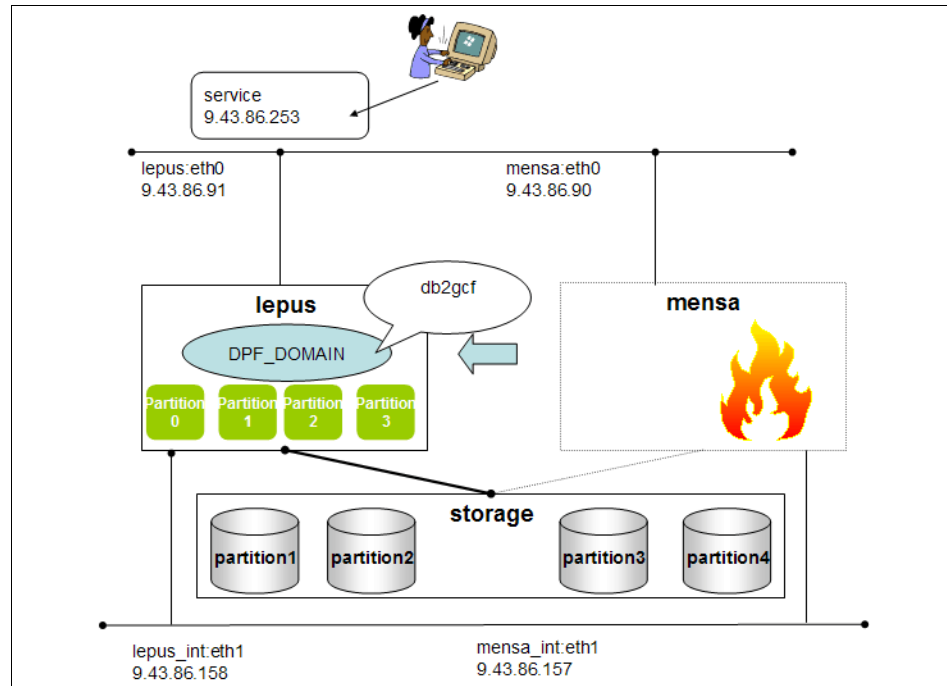


Figure 12-17 Automatic failover to the system pair - node failure

The system states can be seen by issuing the `lssam` command. Example 12-157 provides a sample output. The *Failed Offline* state of all resources on mensa indicates a critical failure.

**Example 12-157** *lssam* sample output

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
|  |- Online IBM.Application:db2_db2inst3_0-rs:lepus
|  ' - Failed offline IBM.Application:db2_db2inst3_0-rs:mensa Node=Offline
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
|  |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus
|  ' - Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa Node=Offline
' - Online IBM.ServiceIP:db2ip_9_43_86_253-rs
|  |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus
|  ' - Failed offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|  |- Online IBM.Application:db2_db2inst3_1-rs:lepus
|  ' - Failed offline IBM.Application:db2_db2inst3_1-rs:mensa Node=Offline
' - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
```

```

        |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
        '- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst3_2-rs
        |- Online IBM.Application:db2_db2inst3_2-rs:lepup
        '- Failed offline IBM.Application:db2_db2inst3_2-rs:mensa Node=Offline
    '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
        |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
        '- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa Node=Offline
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst3_3-rs
        |- Online IBM.Application:db2_db2inst3_3-rs:lepup
        '- Failed offline IBM.Application:db2_db2inst3_3-rs:mensa Node=Offline
    '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
        |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
        '- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa Node=Offline

```

---

When the power is restored on mensa, the cluster manager detects it and changes the resource states on mensa from failed offline to an offline state. Because the DB2 instance owner home directory is on a GPFS that has autoloading turned on, the instance home directory is accessible after powering the mensa ON. If autoloading was turned off, you have to manually start the GPFS subsystem on the node using the following command as user root:

```
mmstartup -a
```

### ***How to find if autoloading is turned on or off***

Run the command `mm1sconfig` and check for the value of the `autoloading` option. If it is *no*, then simply run the following command to turn it on:

```
mmchconfig autoloading=yes
```

Example 12-158 shows the `mm1sconfig` command output in our lab environment.

#### ***Example 12-158 mmlsconfig command output***

---

```

Configuration data for cluster gpfs_cluster.itsosj.sanjose.ibm.com:
-----
clusterName gpfs_cluster.itsosj.sanjose.ibm.com
clusterId 660716716096435580
clusterType lc
autoloading yes
minReleaseLevel 3.2.1.0
dmapiFileHandleSize 32
tiebreakerDisks db2disk1
traceRecycle global
trace trace all 4 tm 2 thread 1 mutex 1 vnode 2 ksvfs 3 klock1 2 io 3 pgallocc 1 mb 1
lock 2 fsck 3
traceFileSize 200000000

```

```
assertOnStructureError yes
tracedevBufferSize 1048576
traceGenSubDir /foobar
```

File systems in cluster gpfs\_cluster.itsosj.sanjose.ibm.com:

```
-----
/dev/db2gpfs
```

---

Example 12-159 shows `Issam` output after `mensa` is powered on.

*Example 12-159 Issam output after mensa powered back on*

---

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
   |- Online IBM.Application:db2_db2inst3_0-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepup
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
   |- Online IBM.Application:db2_db2inst3_1-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
   |- Online IBM.Application:db2_db2inst3_2-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
   |- Online IBM.Application:db2_db2inst3_3-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

There is not any auto-fallback of resources after `mensa` becomes available. The resource groups have to be manually moved using either the `db2haicu move` option or the `rgreq -o move` TSA command.

## Catalog node failure

We turn the power off on the catalog node lepus to simulate the node failure.

All the partition resources and service IP are failed over to mensa and remain online there. The applications receive an SQL30081 code, communication error. Because the automatic client reroute is already set up, the clients establish the database connection using the alternate server information.

Figure 12-18 shows the cluster behavior in the event of a catalog node failure.

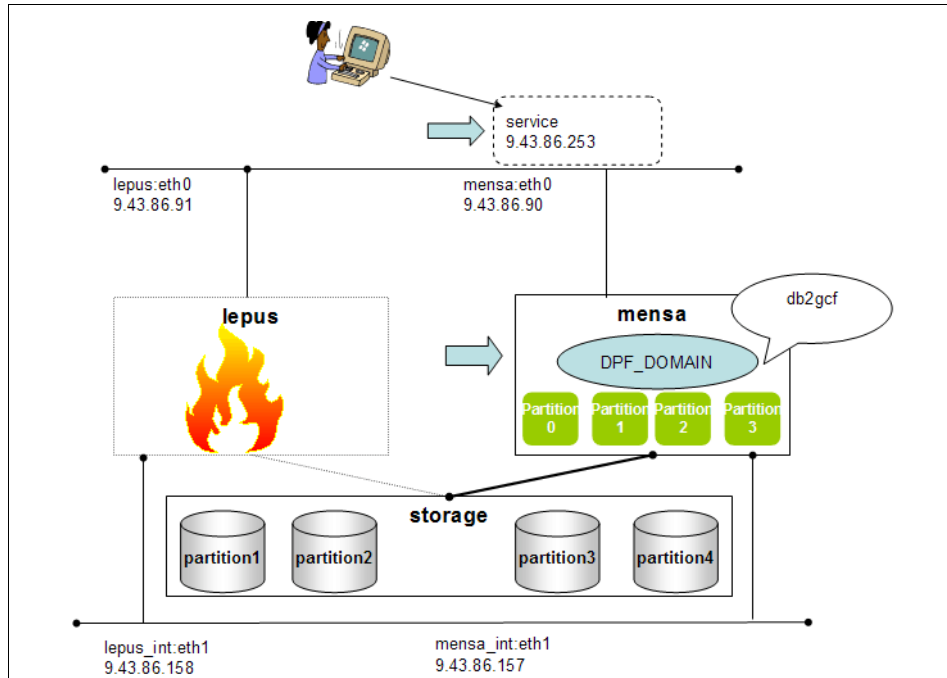


Figure 12-18 Automatic failover to the system pair - catalog node failure

Resources on lepus are in *Failed Offline* state as shown in Example 12-160

Example 12-160 Issam output after lepus powered off

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
  |- Failed offline IBM.Application:db2_db2inst3_0-rs:lepus Node=Offline
  '- Online IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
  |- Failed offline
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus Node=Offline
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
```

```

|- Failed offline IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus Node=Offline
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|- Failed offline IBM.Application:db2_db2inst3_1-rs:lepus Node=Offline
'- Online IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
|- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepus Node=Offline
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
|- Failed offline IBM.Application:db2_db2inst3_2-rs:lepus Node=Offline
'- Online IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
|- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus Node=Offline
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
|- Failed offline IBM.Application:db2_db2inst3_3-rs:lepus Node=Offline
'- Online IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
|- Failed offline
IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus Node=Offline
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa

```

---

When the power is restored on lepus, the cluster manager then changes the *Failed Offline* states to offline states. The DB2 instance owner home directory is accessible immediately after powering lepus ON. System states are shown in Example 12-161.

*Example 12-161 ssam output after power restored on lepus*

---

```

Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
|- Offline IBM.Application:db2_db2inst3_0-rs:lepus
'- Online IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
|- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|- Offline IBM.Application:db2_db2inst3_1-rs:lepus
'- Online IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs

```

```

|- Offline IBM.Application:db2_db2inst3_2-rs:lepus
'- Online IBM.Application:db2_db2inst3_2-rs:mena
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mena
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
|- Offline IBM.Application:db2_db2inst3_3-rs:lepus
'- Online IBM.Application:db2_db2inst3_3-rs:mena
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mena

```

---

## Network failures

In this section we discuss network failures by simulating network interface malfunctions on both nodes.

### ***Private network interface card failure on the machine hosting the catalog partition***

Unplug the eth1 cable (private network cable) connected to lepus and observe the state of the resources. Resources for partition0 and partition1 failed over to mensa along with their mount points and Service IP as expected.

Example 12-162 shows the resources states.

#### *Example 12-162 Issam output after eth1 failure on lepus*

---

```

Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
|- Offline IBM.Application:db2_db2inst3_0-rs:lepus
'- Online IBM.Application:db2_db2inst3_0-rs:mena
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mena
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
|- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:mena
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
|- Offline IBM.Application:db2_db2inst3_1-rs:lepus
'- Online IBM.Application:db2_db2inst3_1-rs:mena
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mena
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
|- Offline IBM.Application:db2_db2inst3_2-rs:lepus
'- Online IBM.Application:db2_db2inst3_2-rs:mena
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
|- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mena

```

```

Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
   |- Offline IBM.Application:db2_db2inst3_3-rs:lepus
   '- Online IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa

```

---

Plug eth1 back on lepus, and there should not be any auto-failback. You can use either the **db2haicu move** option or the **rgreq TSA** command to move the resource groups back to the preferred node.

### ***Private network interface card failure on the machine hosting the data partitions***

Unplug the eth1 cable from mensa. There was a brief period during which the resources for partition2 and partition3 were in a *Pending Offline* state, but then were failed over to lepus and remained *Online* there.

Example 12-163 shows the state of partition resources.

#### ***Example 12-163 Issam output after eth1 failure on mensa***

---

```

Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
   |- Online IBM.Application:db2_db2inst3_0-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
   |- Online IBM.Application:db2_db2inst3_1-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepus
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
   |- Online IBM.Application:db2_db2inst3_2-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
   |- Online IBM.Application:db2_db2inst3_3-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs

```



```
| - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
'- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

Plug eth1 back on mensa, and there should not be any auto-failback. You can use either the **db2haicu** or the **rgreq** TSA command to move the resource groups back to the preferred node.

## Disk I/O failures

We discuss disk I/O failures in this section by simulating disk controller malfunctions on the both nodes.

Gently pull the fiber-optic cable from the RAID controller on mensa. There was a brief period through which the mount point resources were in an *Unknown* state. Then the partition and mount point resources were failed over to lepus and remain *Online* there. Resource states are shown in Example 12-164.

### Example 12-164 Issam output after disk failure on mensa

---

```
Online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
  | - Online IBM.Application:db2_db2inst3_0-rs
    | - Online IBM.Application:db2_db2inst3_0-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
  | - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
    | - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
  '- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
    | - Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepup
    '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
  | - Online IBM.Application:db2_db2inst3_1-rs
    | - Online IBM.Application:db2_db2inst3_1-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
    | - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
  | - Online IBM.Application:db2_db2inst3_2-rs
    | - Online IBM.Application:db2_db2inst3_2-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_2-rs:mensa
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
    | - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
  | - Online IBM.Application:db2_db2inst3_3-rs
    | - Online IBM.Application:db2_db2inst3_3-rs:lepup
    '- Offline IBM.Application:db2_db2inst3_3-rs:mensa
  '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
    | - Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
    '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

To move the partitions back to mensa, connect the fiber-optic cable on mensa and initiate a move request using **db2haicu** maintenance mode. The same behaviors applied when the disk failure test was performed on lepus.

## DB2 failures

Here we observe how the resource states change when there is a failure on the DB2 instance.

When an instance crashes, SA MP detects and restarts the instance. Partition resources are in *Pending Online* state for a brief period, then move to the *Online* state. Example 12-165 shows the resources states during an instance crash.

### *Example 12-165 Resource states during an instance crash*

---

```

Pending online IBM.ResourceGroup:db2_db2inst3_0-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst3_0-rs
   |- Pending online IBM.Application:db2_db2inst3_0-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepus
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepus
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst3_1-rs
   |- Pending online IBM.Application:db2_db2inst3_1-rs:lepus
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepus
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst3_2-rs
   |- Offline IBM.Application:db2_db2inst3_2-rs:lepus
   '- Pending online IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Pending Online IBM.Application:db2_db2inst3_3-rs
   |- Offline IBM.Application:db2_db2inst3_3-rs:lepus
   '- Online IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa

```

---

## 12.7.5 Maintenance

For a regular maintenance, you might need to stop and start the instance. When an instance is stopped using the **db2stop** command, the cluster manager locks the instance resource group. The resource group is not brought offline because that would cause the cluster manager to unmount the file systems associated with the mount resources in the resource group. Note that any failure on the node does not cause the cluster manager to initiate a failover, because the resource group is in *Lock* state.

Example 12-166 shows the resources states after stopping the instance.

### *Example 12-166 Resource states after db2stop*

---

```
Pending online IBM.ResourceGroup:db2_db2inst3_0-rg Request=Lock Nominal=Online
|- Offline IBM.Application:db2_db2inst3_0-rs
   |- Offline IBM.Application:db2_db2inst3_0-rs:lepuc
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepuc
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepuc
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst3_1-rg Request=Lock Nominal=Online
|- Offline IBM.Application:db2_db2inst3_1-rs
   |- Offline IBM.Application:db2_db2inst3_1-rs:lepuc
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepuc
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst3_2-rg Request=Lock Nominal=Online
|- Offline IBM.Application:db2_db2inst3_2-rs
   |- Offline IBM.Application:db2_db2inst3_2-rs:lepuc
   '- Offline IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepuc
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Pending online IBM.ResourceGroup:db2_db2inst3_3-rg Request=Lock Nominal=Online
|- Offline IBM.Application:db2_db2inst3_3-rs
   |- Offline IBM.Application:db2_db2inst3_3-rs:lepuc
   '- Offline IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepuc
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

The instance resource group remains “locked” until a successful **db2start** command is issued. Start the DB2 instance manually by issuing the **db2start** command; when it completes, the lock is removed. Henceforth, the cluster manager detects any failures, and the instance resource group is either failed over or restarted (as warranted) by the cluster manager.

## db2haicu maintenance mode

When a system is already configured for High Availability, **db2haicu** runs in maintenance mode. Entering **db2haicu** on the node produces the menu shown in Example 12-167. This menu can be used to carry out various maintenance tasks and change any cluster-manager-specific, DB2-specific, or network-specific values configured during the initial setup. Any time you want to close **db2haicu** without entering any value, you can do so by pressing Ctrl+C.

### *Example 12-167 db2haicu maintenance mode*

---

Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst3. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

When you use db2haicu to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with db2haicu' in the DB2 Information Center. db2haicu is searching the current machine for an existing active cluster domain ...

db2haicu found a cluster domain called DPF\_DOMAIN on this machine. The cluster configuration that follows will apply to this domain.

Select an administrative task by number from the list below:

1. Add or remove cluster nodes.
  2. Add or remove a network interface.
  3. Add or remove DB2 database partitions.
  4. Add or remove a highly available database.
  5. Add or remove a mount point.
  6. Add or remove an IP address.
  7. Add or remove a non-critical path.
  8. Move DB2 database partitions and HADR databases for scheduled maintenance.
  9. Change failover policy for this instance.
  10. Create a new quorum device for the domain.
  11. Destroy the domain.
  12. Exit.
- 

For some maintenance, if you want to move database partitions from mensa to lepus, you can choose the administrative task 8 as shown in Example 12-168. Keep in mind that clients experience an outage from doing this process.

### Example 12-168 Partitions move from mensa

---

```
Enter your selection:
8
Do you want to review the status of each cluster node in the domain before you begin?
[1]
1. Yes
2. No
1
Domain Name: DPF_DOMAIN
  Node Name: lepus --- State: Online
  Node Name: mensa --- State: Online
Enter the name of the node away from which DB2 partitions and HADR primary roles should
be moved:
mensa
Cluster node mensa hosts the following resources associated with the database manager
instance db2inst3:
  DB2 database partition number 3
  DB2 database partition number 2
All of these cluster resource groups will be moved away from the cluster node mensa.
This will take their database partitions offline for the duration of the move, or cause
an HADR role switch for HADR databases. Clients will experience an outage from this
process. Are you sure you want to continue? [1]
1. Yes
2. No
1
Submitting move request for resource group db2_db2inst3_3-rg ...
The move request for resource group db2_db2inst3_3-rg was submitted successfully.
Submitting move request for resource group db2_db2inst3_2-rg ...
The move request for resource group db2_db2inst3_2-rg was submitted successfully.
Do you want to make any other changes to the cluster configuration? [1]
1. Yes
2. No
2
All cluster configurations have been completed successfully. db2haicu exiting
...
```

---

You then see the resources for partition2 and partition3 online on lepus as shown in Example 12-169.

### Example 12-169 Resource for partitions 2 and 3 moved to lepus

---

```
Online IBM.ResourceGroup:db2_db2inst3_2-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
  |- Online IBM.Application:db2_db2inst3_2-rs:lepus
  '- Offline IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
  |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepus
  '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
  |- Online IBM.Application:db2_db2inst3_3-rs:lepus
  '- Offline IBM.Application:db2_db2inst3_3-rs:mensa
```

```
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
  |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepus
  '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa
```

---

Now to move database partition2 and partition3 back to mensa, you can use the option 8 and selectively choose the partitions by answering **NO** to the initial prompt as shown in Example 12-170. If you try to invoke **db2haicu** from mensa, it fails with an SQL1032N error because it does not find any database partitions that are currently online on the cluster node mensa. So you can run **db2haicu** from lepus, which hosts all the database partitions at this point.

#### *Example 12-170 Partitions move back to mensa*

---

```
Enter your selection:
8
Do you want to review the status of each cluster node in the domain before you begin?
[1]
1. Yes
2. No

Domain Name: DPF_DOMAIN
  Node Name: lepus --- State: Online
  Node Name: mensa --- State: Online
Enter the name of the node away from which DB2 partitions and HADR primary roles should
be moved:
lepus
Cluster node lepus hosts the following resources associated with the database manager
instance db2inst3:
  DB2 database partition number 3
  DB2 database partition number 2
  DB2 database partition number 1
  DB2 database partition number 0
All of these cluster resource groups will be moved away from the cluster node lepus.
This will take their database partitions offline for the duration of the move, or cause
an HADR role switch for HADR databases. Clients will experience an outage from this
process. Are you sure you want to continue? [1]
1. Yes
2. No
2
  DB2 database partition number 2
1. Yes
2. No
1
Submitting move request for resource group db2_db2inst3_2-rg ...
The move request for resource group db2_db2inst3_2-rg was submitted successfully.
  DB2 database partition number 1
1. Yes
2. No
2
  DB2 database partition number 3
1. Yes
2. No
```

```
1
Submitting move request for resource group db2_db2inst3_3-rg ...
The move request for resource group db2_db2inst3_3-rg was submitted successfully.
Do you want to make any other changes to the cluster configuration? [1]
1. Yes
2. No
2
All cluster configurations have been completed successfully. db2haicu exiting
...
```

---

You will see resources for partition2 and partition3 fail back on mensa.

## Disabling High Availability

To disable the HA configuration for a particular instance, the **db2haicu -disable** command can be used. After issuing this command, the system does not respond to any failures, and all resource groups for the instance are locked as shown in Example 12-171. Any maintenance work can be performed in this state without risk of cluster manager intervention.

### *Example 12-171 Disabling High Availability for a DPF instance*

---

Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is db2inst3. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths

...

Are you sure you want to disable high availability (HA) for the database instance db2inst3. This will lock all the resource groups for the instance and disable the HA configuration parameter. The instance will not failover if a system outage occurs while the instance is disabled. You will need to run db2haicu again to enable the instance for HA. Disable HA for the instance db2inst3? [1]

1. Yes

2. No

1

Disabling high availability for instance db2inst3 ...

Locking the resource group for DB2 database partition 3 ...

Locking the resource group for DB2 database partition 3 was successful.

Locking the resource group for DB2 database partition 2 ...

Locking the resource group for DB2 database partition 2 was successful.

Locking the resource group for DB2 database partition 1 ...

```

Locking the resource group for DB2 database partition 1 was successful.
Locking the resource group for DB2 database partition 0 ...
Locking the resource group for DB2 database partition 0 was successful.
Disabling high availability for instance db2inst3 was successful.
All cluster configurations have been completed successfully. db2haicu exiting
...

```

---

Example 12-172 shows the locked state for all the resource groups.

*Example 12-172 Resource groups locked after disabling HA*

---

```

Online IBM.ResourceGroup:db2_db2inst3_0-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst3_0-rs
   |- Online IBM.Application:db2_db2inst3_0-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_0-rs:mensa
|- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0000-rs:mensa
'- Online IBM.ServiceIP:db2ip_9_43_86_253-rs
   |- Online IBM.ServiceIP:db2ip_9_43_86_253-rs:lepup
   '- Offline IBM.ServiceIP:db2ip_9_43_86_253-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_1-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst3_1-rs
   |- Online IBM.Application:db2_db2inst3_1-rs:lepup
   '- Offline IBM.Application:db2_db2inst3_1-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs
   |- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:lepup
   '- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0001-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_2-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst3_2-rs
   |- Offline IBM.Application:db2_db2inst3_2-rs:lepup
   '- Online IBM.Application:db2_db2inst3_2-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:lepup
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0002-rs:mensa
Online IBM.ResourceGroup:db2_db2inst3_3-rg Request=Lock Nominal=Online
|- Online IBM.Application:db2_db2inst3_3-rs
   |- Offline IBM.Application:db2_db2inst3_3-rs:lepup
   '- Online IBM.Application:db2_db2inst3_3-rs:mensa
'- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs
   |- Offline IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:lepup
   '- Online IBM.Application:db2mnt-db2data_db2inst3_NODE0003-rs:mensa

```

---

To enable HA, just issue the **db2haicu** command again, and select **Yes** when prompted to continue.



## The `db2haicu -delete` option

`db2haicu` can also be run with the `-delete` option. This option removes a system's entire HA configuration and deletes all resources in the cluster for the instance in question. If no other instance is using the domain at the time, the domain is deleted as well.

We recommend that you run `db2haicu` with the `-delete` option on an instance before it is made highly available. This makes sure that you are starting from scratch and not building on top of leftover resources.

For example, when running `db2haicu` with an XML file, any invalid attribute in the file causes `db2haicu` to exit with a non-zero error code. However, before `db2haicu` is run again with the corrected XML file, you can run the `-delete` option to make sure that any temporary resources created during the initial run are cleaned up.





## Q replication

In this chapter we provide an introduction to Q replication and a simple overview of its structure. We also give you step-by-step instructions for setting up a unidirectional Q replication.

## 13.1 Introduction

Q replication is a replication solution that uses WebSphere MQ message queues to transmit transactions between source and target databases.

Q replication is divided into three types:

▶ Unidirectional:

In *unidirectional Q replication*, changes are captured at the source and replicated to the target. Unidirectional Q replication can happen from one source to one target or many targets. In unidirectional replication, typically the target is used for read-only.

▶ Bidirectional:

In *bidirectional Q replication*, two tables on two servers replicate each other. Changes made on either table are replicated to the corresponding table. You cannot replicate a subset of rolls, and each table must have the same number of columns and data types for those columns, though you can have different schema and table names.

▶ Peer-to-peer:

In *peer-to-peer Q replication*, tables are replicated between two or more servers. As in bidirectional Q replication, the replicated tables must all be of the same structure, though each table can have its own schema and table name.

Q replication can be set up in many different configurations. You can use Q replication for replication between databases on the same server or remote servers. You can set up a one-to-many relationship or many-to-one relationship.

For more details on different Q replication configurations, refer to the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

Next we describe the basic structure of Q replication:

- ▶ A Q Capture program that runs on the source server reads the DB2 recovery log for changes to the source tables specified for replication. It then places the messages in a queue called *send queue*. The Q Capture program uses a set of DB2 tables called *Q Capture Control tables*. The capture control tables store information about the replication sources, corresponding targets, MQ queues being used, and other data.
- ▶ Q subscriptions define the source and target tables to be replicated. Q subscriptions need a replication queue map to identify the WebSphere MQ queues used for sending and receiving transactions.

- ▶ The Q Apply program runs on the target server, receiving the messages and applying the transactions to the target tables. The Q Apply program also has a set of DB2 tables called *Q Apply Control tables*.

In the rest of this chapter we focus on a unidirectional Q replication setup.

## 13.2 Unidirectional setup

This section provides a step-by-step example of a unidirectional Q replication setup between two remote servers. See Figure 13-1.

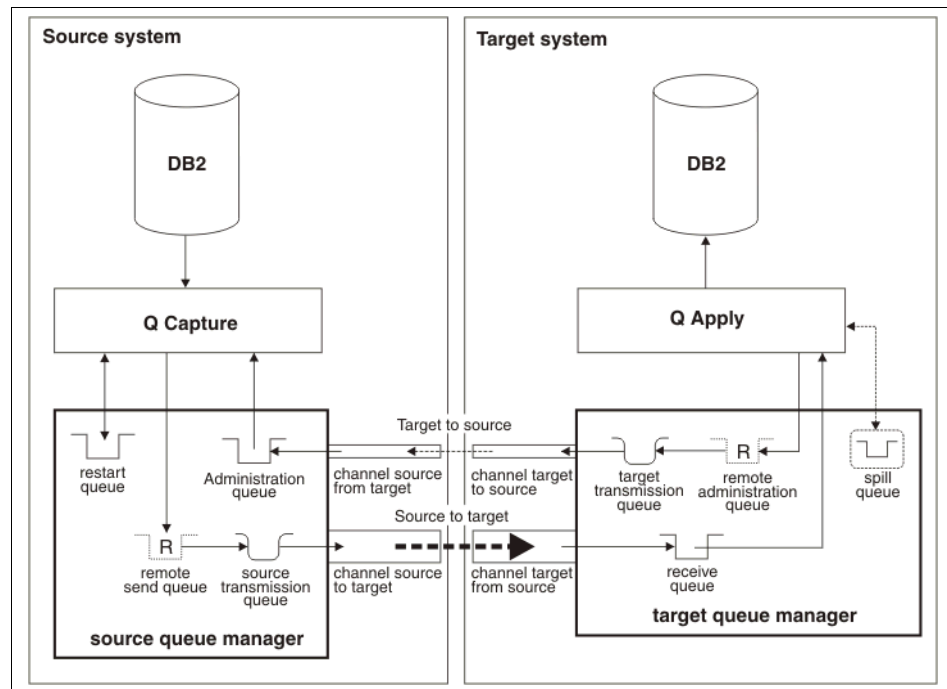


Figure 13-1 Unidirectional setup diagram

The following tasks must be completed for setting up a unidirection Q replication:

1. Set up the database on both the source and target database servers.
2. Set up WebSphere MQ objects on the source server and the target server.
3. Start the listener and the channel on the source and the target.
4. Optional: Test the queues.
5. Create the Q Capture Control tables.
6. Create the Q Apply Control tables.
7. Create a Q subscription.

8. Configure target to source server connectivity.
9. Start Q Capture.
10. Start Q Apply.

In our example, we used two Windows servers running DB2 Version 9.1 and WebSphere MQ 6.0. The step-by-step process to set up a unidirectional Q replication with remote servers is as follows:

1. Initial database setup on the source and the target server:
  - a. The source database (SAMPLE in our test case) must have the database configuration parameter LOGRETAIN set to RECOVERY.
 

```
db2 update database configuration for sample using LOGRETAIN
RECOVERY
```

A backup is required if you are setting the LOGRETAIN to recovery for the first time.
  - b. Identify your target database. For our example, the target database is TGTDB.
2. Setup of WebSphere MQ objects on the source server:
  - a. Create a queue manager QMGR1 from the command window:
 

```
crtmqm QMGR1
```
  - b. Start the Q manager that you have just created:
 

```
strmqm QMGR1
```
  - c. To create the needed MQ objects, run the following command with the example script shown in Example 13-1. Remember to replace the IP and port number with your required IP and port number:
 

```
runmqsc QMGR1 < QMGR1.DEF
```

*Example 13-1 QMGR1.DEF*

---

```
**This script creates the following objects within queue manager QMGR1
**at the source:

**A local administration queue
DEF QLOCAL('SAMPLE_ADMINQ') DESCR('Capture Admin Queue') REPLACE

**A local restart queue
DEF QLOCAL('SAMPLE_RESTARTQ') DESCR('Capture Restart Queue') REPLACE

**A remote queue definition for Q Capture to put data messages
DEF QREMOTE('SAMPLE2TGTDQB') XMITQ('QMGR2') RNAME('SAMPLE2TGTDQB')
RQMNAME('QMGR2') REPLACE
```

```

**A local transmission queue
DEF QLOCAL('QMGR2') USAGE(XMITQ) REPLACE

**A sender channel from QMGR1 to queue manager QMGR2 using transmission
**queue
**Replace the ip with your ip and the port number with your port number
DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(SDR) TRPTYPE(TCP)
CONNNAME('9.43.86.78 (1416)') XMITQ('QMGR2') REPLACE

**A receiver channel from QMGR2
DEF CHANNEL('QMGR2.TO.QMGR1') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE

```

---

### 3. Setup of WebSphere MQ objects on the target server:

- a. Create a queue manager with a different name as the source from the command window:

```
crtmqm QMGR2
```

- b. Start the Q manager that you have just created:

```
strmqm QMGR2
```

- c. To create the needed MQ objects, run the following command with the example script shown in Example 13-2. Remember to replace the IP and port number with your required IP and port number:

```
runmqsc QMGR2 < QMGR2.DEF
```

#### *Example 13-2 QMGR2.DEF*

---

```

**This script creates the following objects within queue manager QMGR2
**at the target:

```

```

**A remote queue definition that points to the administration queue at
**the source

```

```

DEF QREMOTE('SAMPLE_ADMINQ') XMITQ('QMGR1') +
RNAME('SAMPLE_ADMINQ') RQMNAME('QMGR1') REPLACE

```

```

**A model queue definition for spill queues that hold data messages
**from Q Capture during the load

```

```

DEF QMODEL('IBMQREP.SPILL.MODELQ') DEFSOPT(SHARED) +
MAXDEPTH(500000) MSGDLVSQ(FIFO) DEFTYPE(PERMDYN) REPLACE

```

```

**A local queue for Q Apply to get data messages from the Q Capture
**program at the source

```

```

DEF QLOCAL('SAMPLE2TGTDQB') REPLACE

```

```

**A local transmission queue
DEF QLOCAL('QMGR1') USAGE(XMITQ) REPLACE

**A sender channel from QM2 to queue manager QMGR1 using transmission
**queue QMGR1
**Replace the ip with your ip and the port number with your port number
DEF CHANNEL('QMGR2.TO.QMGR1') CHLTYPE(SDR) TRPTYPE(TCP) +
CONNAME('9.43.86.75 (1415)') XMITQ('QMGR1') REPLACE

**A receiver channel from QMGR1
DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE

```

---

4. Start both the listener and channel on the source and the target:

- a. On the source, from a command window, start the listener:

```
RUNMQLSR -T TCP -M QMGR1 -P 1415
```

- b. On the target, from a command window, start the listener:

```
RUNMQLSR -T TCP -M QMGR2 -P 1416
```

- c. On the source, from a command window, start the channel:

To start the channel, first use the RUNMQSC command to start the interactive MQSC command line:

```
RUNMQSC QMGR1
```

This command starts the channel:

```
START CHANNEL(QMGR1.TO.QMGR2)
```

This command ends the `runmqsc` session.

```
END
```

- d. On the target from a command window, start the channel:

```
RUNMQSC QMGR2
```

```
START CHANNEL(QMGR2.TO.QMGR1)
```

```
END
```



5. Optional: Test the queues:

We test the message queue setup by putting a message from the source system and trying to get the message from the target system.

- a. In the source, we put a message, Test message to send, into the message queue as shown in Example 13-3.
  - i. Press Enter to send the message.
  - ii. Press Enter to end the prompt.

*Example 13-3 amqsput*

---

```
>amqsput SAMPLE2TGTDDBQ QMGR1
Sample AMQSPUTO start
target queue is SAMPLE2TGTDDBQ
Test message to send
Sample AMQSPUTO end
```

---

- b. Example 13-4 shows how to receive the message from the target. Press Enter to end the prompt.

*Example 13-4 amqsget*

---

```
>amqsget SAMPLE2TGTDDBQ QMGR2
Sample AMQSGETO start
message <Test message to send>
no more messages
Sample AMQSGETO end
```

---

6. Creating the Q Capture Control tables:
  - a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box. See Figure 13-2.

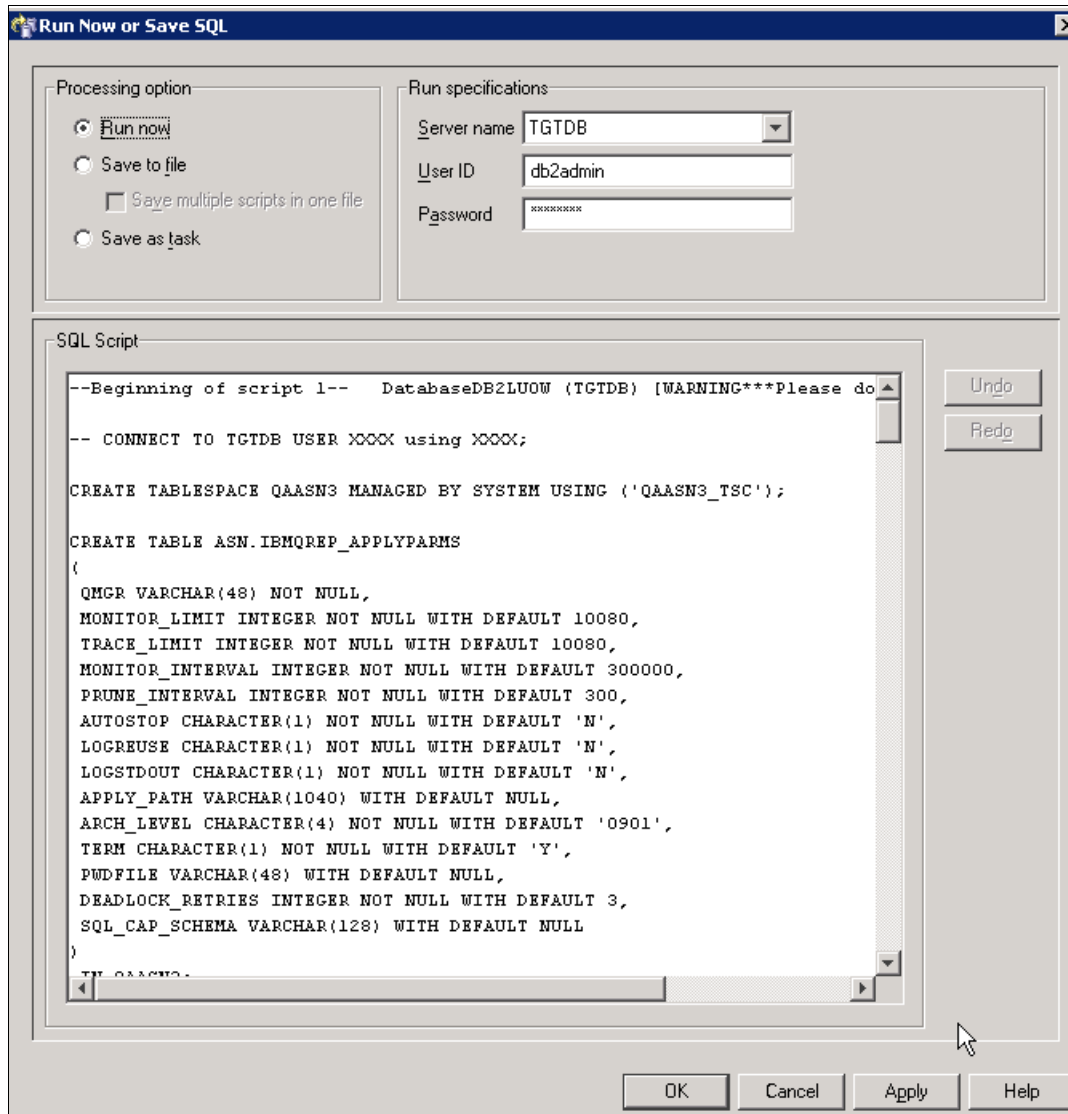


Figure 13-2 Replication Center Launchpad

- b. Click **1. Create Q Capture Control Tables** from the Replication Center Launchpad. This opens the Create Q Capture Control Tables Wizard. From the first panel in the wizard, choose **Typical** or **Custom**, then click **Next**. See Figure 13-3.

**Note:** When creating the Q Capture Control tables, choosing the Custom option allows you to define the tablespace or tablespaces that you want the tables to reside in.

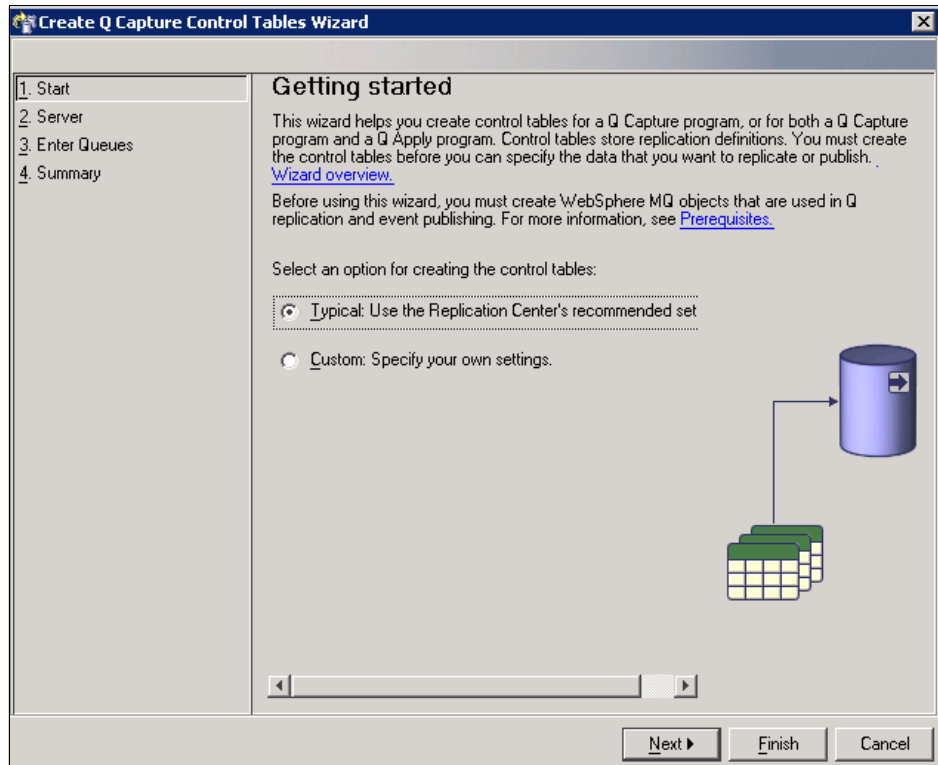


Figure 13-3 Create Q Capture Control Tables Wizard, panel 1

- c. Select the Q Capture server from the list (source server). This should populate the User ID, Password, and Q Capture schema. The default schema is ASN, but can be changed if necessary. Click **Next** to proceed. See Figure 13-4.

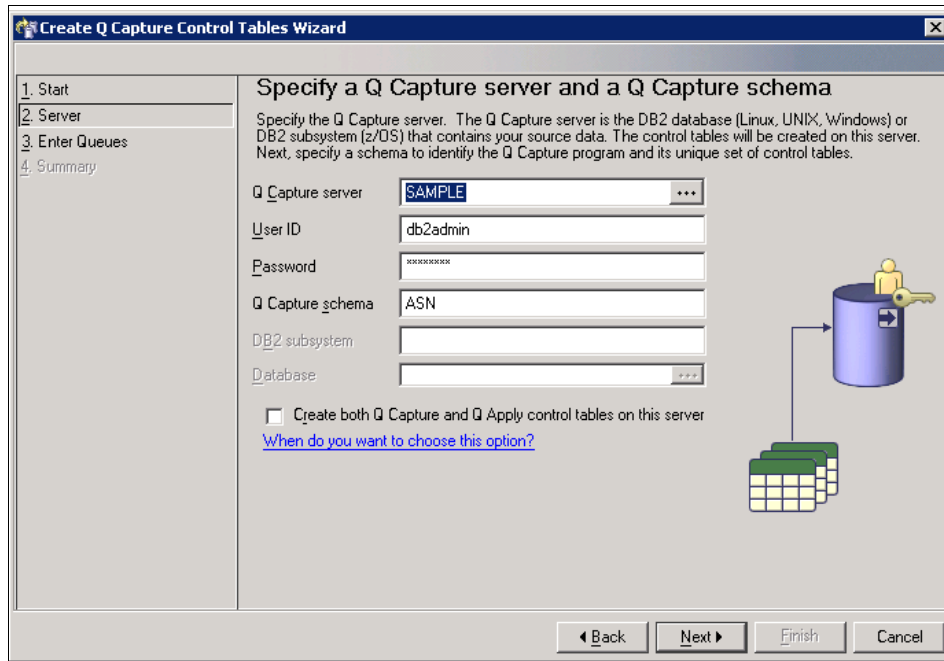


Figure 13-4 Create Q Capture Control Tables Wizard, panel 2

- d. Enter the Queue manager name for the source server. In our example, the Queue manager is QMGR1. Enter the Administration queue and Restart queue, which are SAMPLE\_ADMINQ and SAMPLE\_RESTARTQ respectively in our example. Click **Next** to proceed to the summary panel. See Figure 13-5.

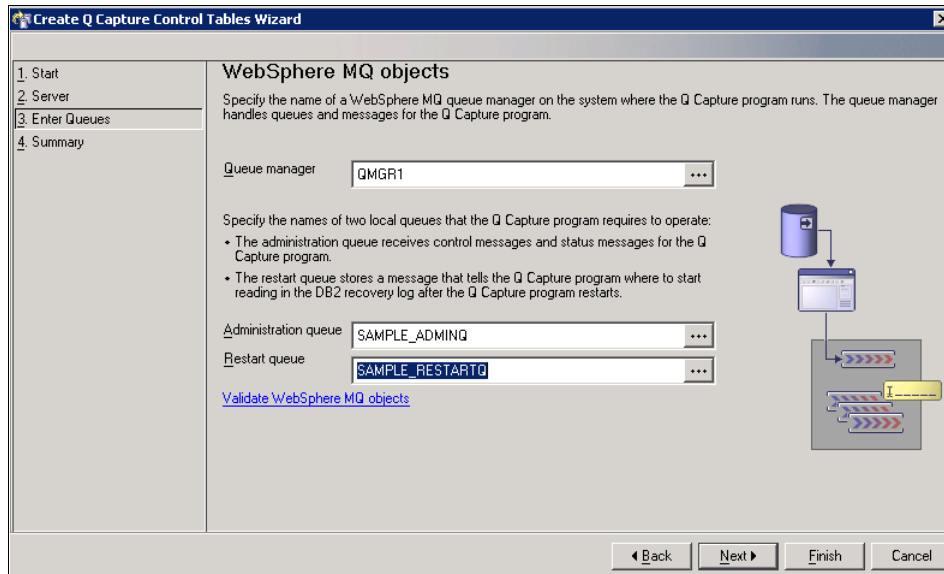


Figure 13-5 Create Q Capture Control Tables Wizard, panel 3

e. Review the Summary panel and click **Finish**. See Figure 13-6.

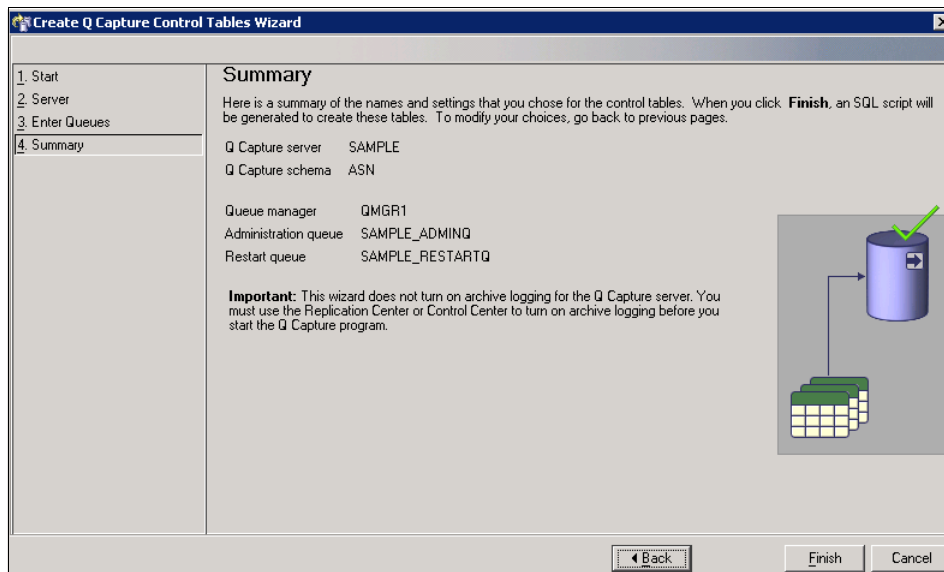


Figure 13-6 Create Q Capture Control Tables Wizard: Summary

- f. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 13-7. After saving the SQL, select **Run now** and click **OK**.

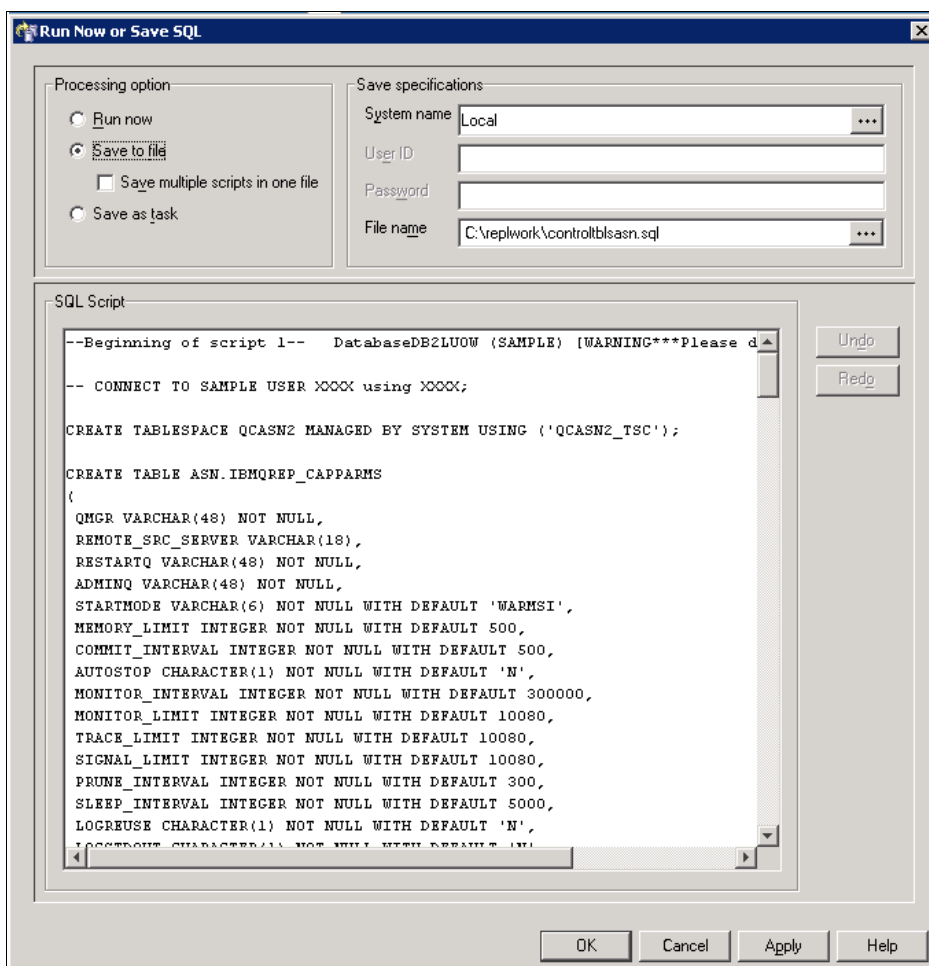


Figure 13-7 Run Now or Save SQL panel for Q Capture Control Tables

## 7. Creating the Q Apply Control tables.

This can be done from the target or source server. If done from the source, make sure that the target server is cataloged in the source.

- a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box.
- b. Click **2. Create Q Apply Control Tables** from the Replication Center Launchpad. This opens the Create Q Apply Control Tables Wizard. From the first panel in the wizard, choose **Typical** or **Custom**, then click **Next**. See Figure 13-8.

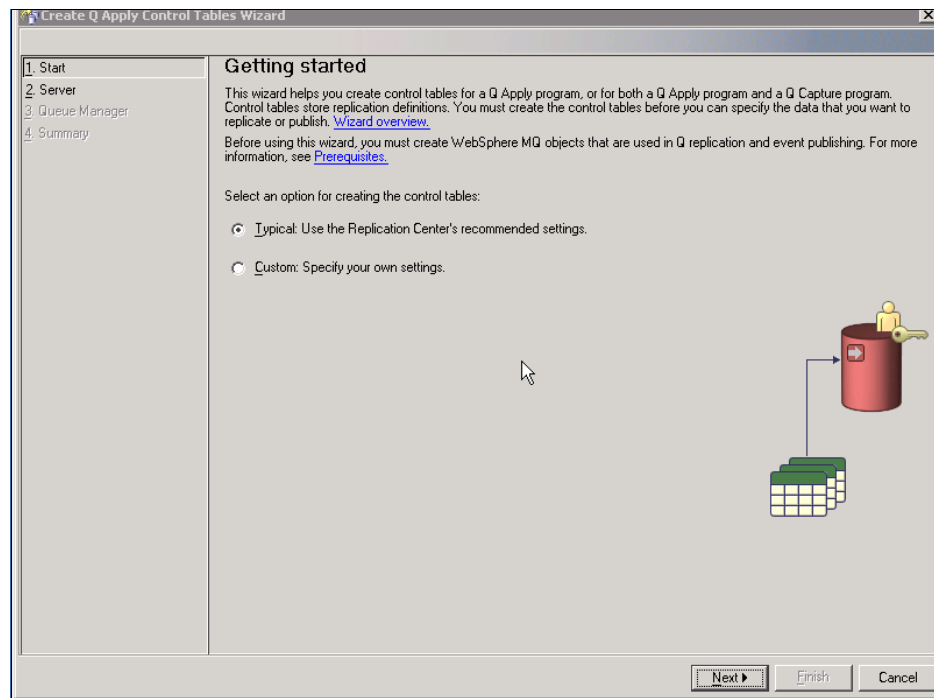


Figure 13-8 Create Q Apply Control Tables Wizard - Getting started

- c. Select the Q Apply server from the list (target server). This should populate the User ID, Password, and Q Apply schema. The default schema is ASN, but can be changed if necessary. Click **Next** to proceed. See Figure 13-9.

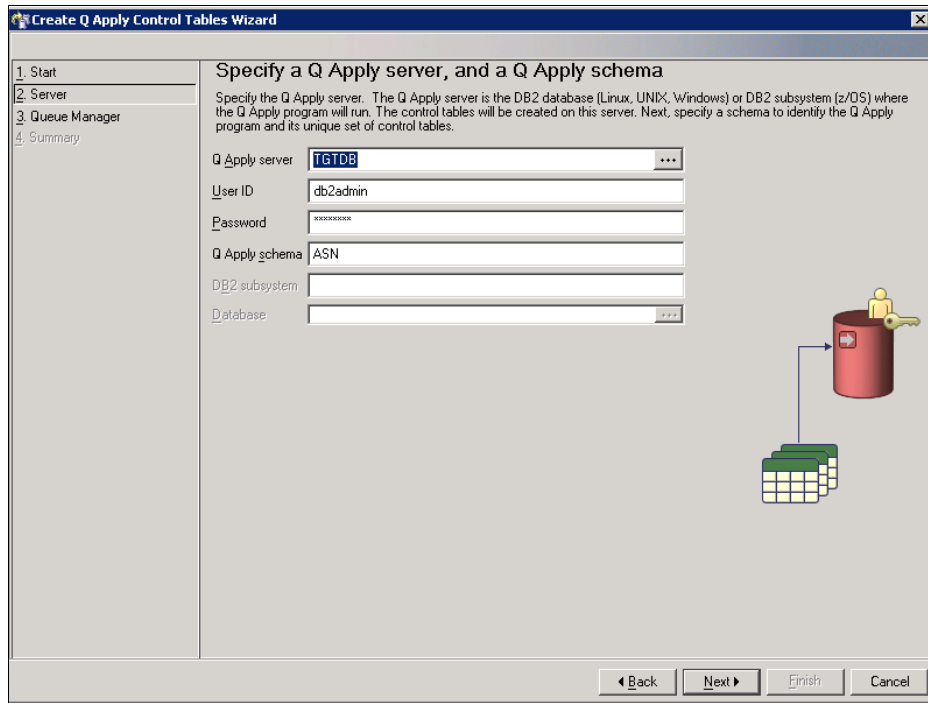


Figure 13-9 Create Q Apply Control Tables Wizard panel 2



- d. Insert the Queue manager name for the target server. In our example, the Queue manager is QMGR2. Click **Next** to proceed to the summary panel. See Figure 13-10.

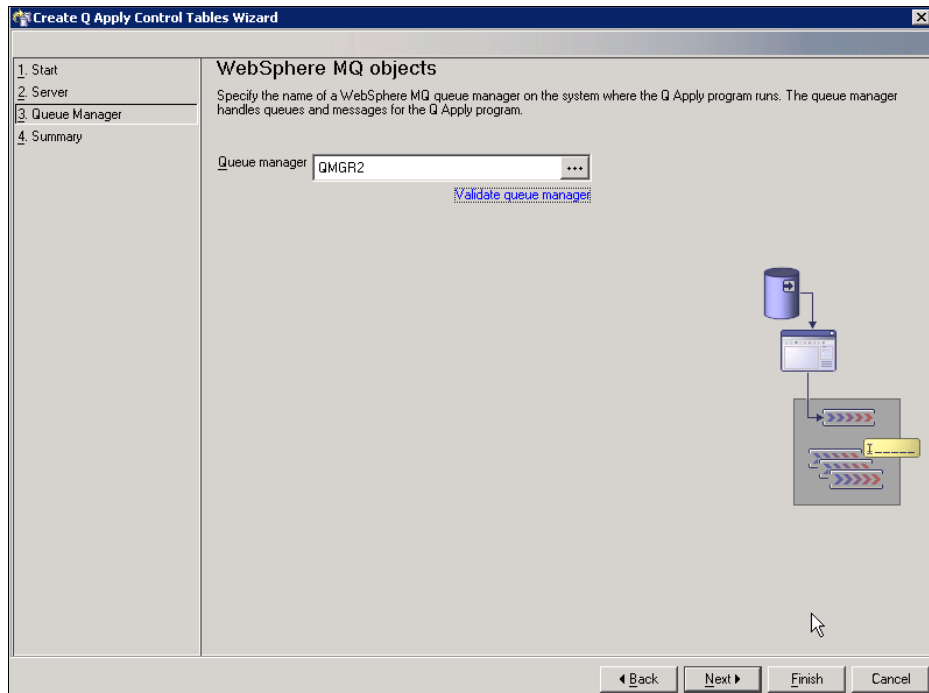


Figure 13-10 Create Q Apply Control Tables Wizard panel 3

e. Review the Summary panel and click **Finish**. See Figure 13-11.

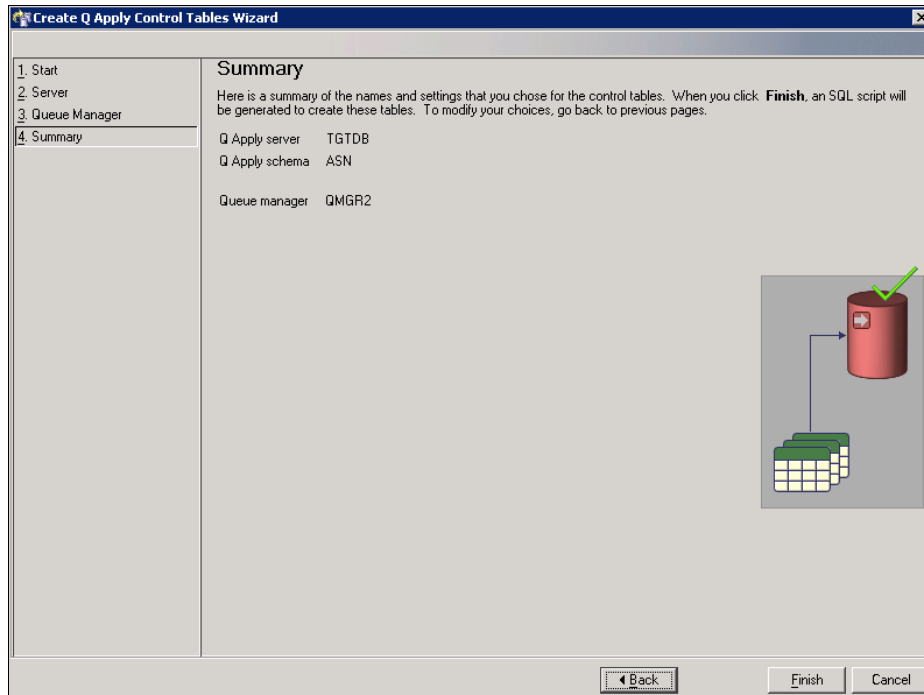


Figure 13-11 Create Q Apply Control Tables Wizard panel 4

- f. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 13-12. After saving the SQL, select **Run now** and click **OK**.

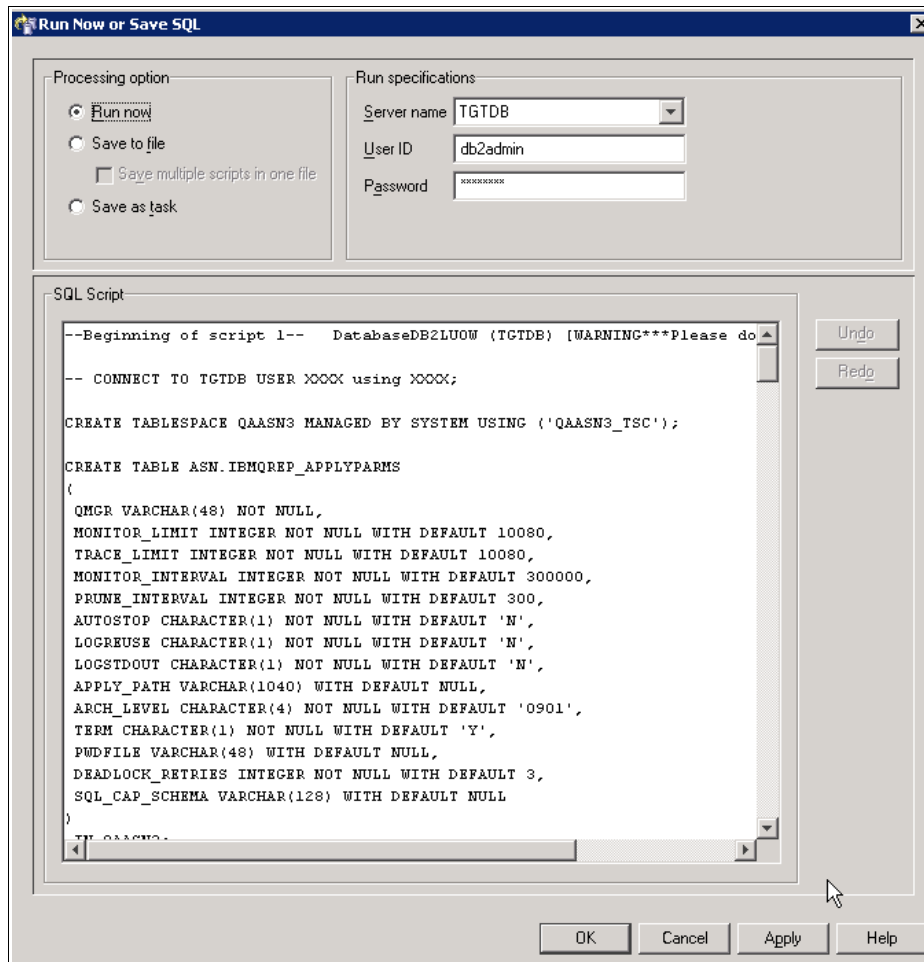


Figure 13-12 Run Now or Save SQL panel for Q Apply Control Tables

8. Create a Q subscription:
  - a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box.

- b. Click **3. Create a Q Subscription** from the Replication Center Launchpad. This opens the Create a Q Subscription Wizard. From the first panel in the wizard, click **Next**. See Figure 13-13.

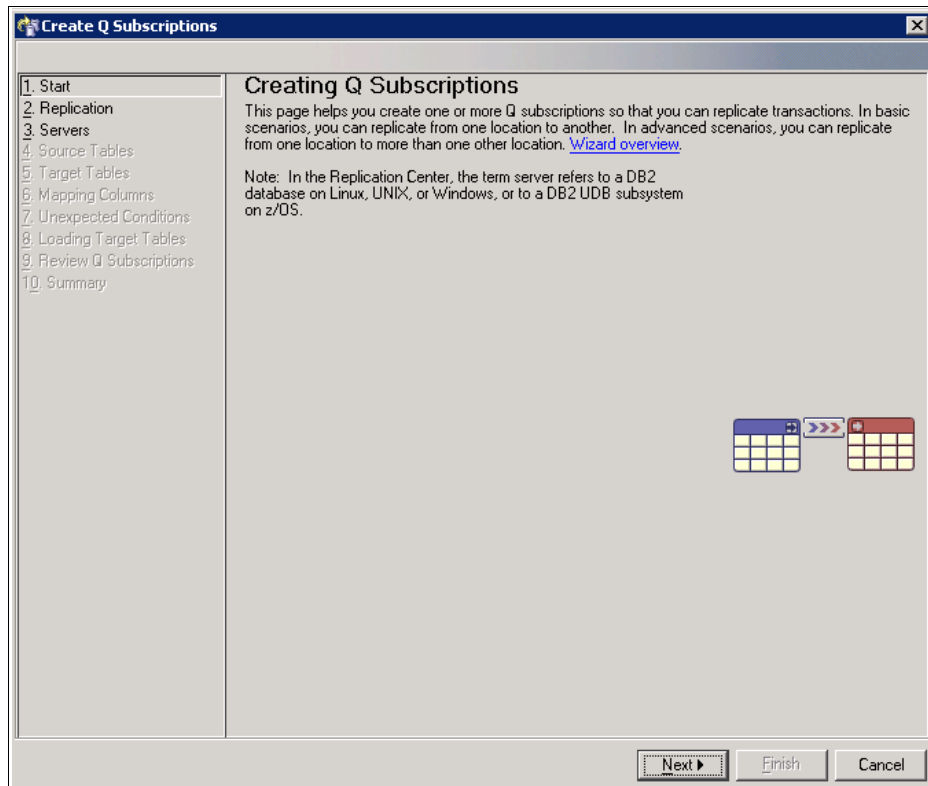


Figure 13-13 Create Q Subscription Wizard Start

- c. Choose the type of Q replication you require. Our example is for unidirectional replication. See Figure 13-14. If you choose anything other than unidirectional, additional MQ setup steps might be required. See DB2V9 Information Center for more information at:

[http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.nav.doc/navcontainers/iypnavc\\_config\\_q\\_replication.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.nav.doc/navcontainers/iypnavc_config_q_replication.html)

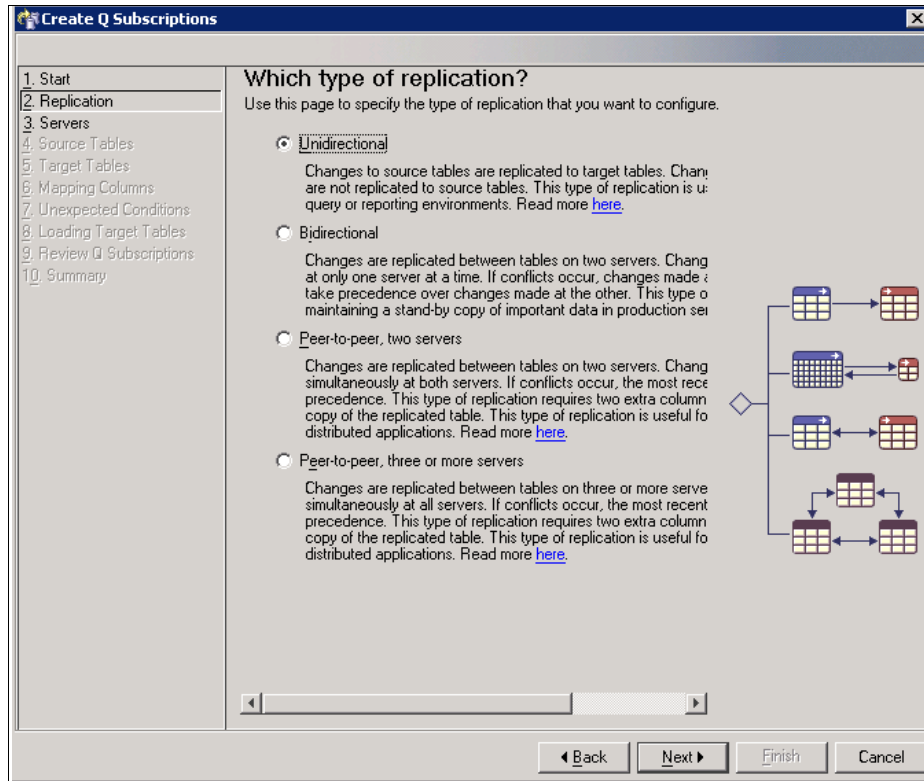


Figure 13-14 Create Q Subscription Wizard Replication

- d. Choose the Source and Target servers and schema if not already populated. In the Queues section, click the button next to the box to open the Select Replication Queue Map window. See Figure 13-15.

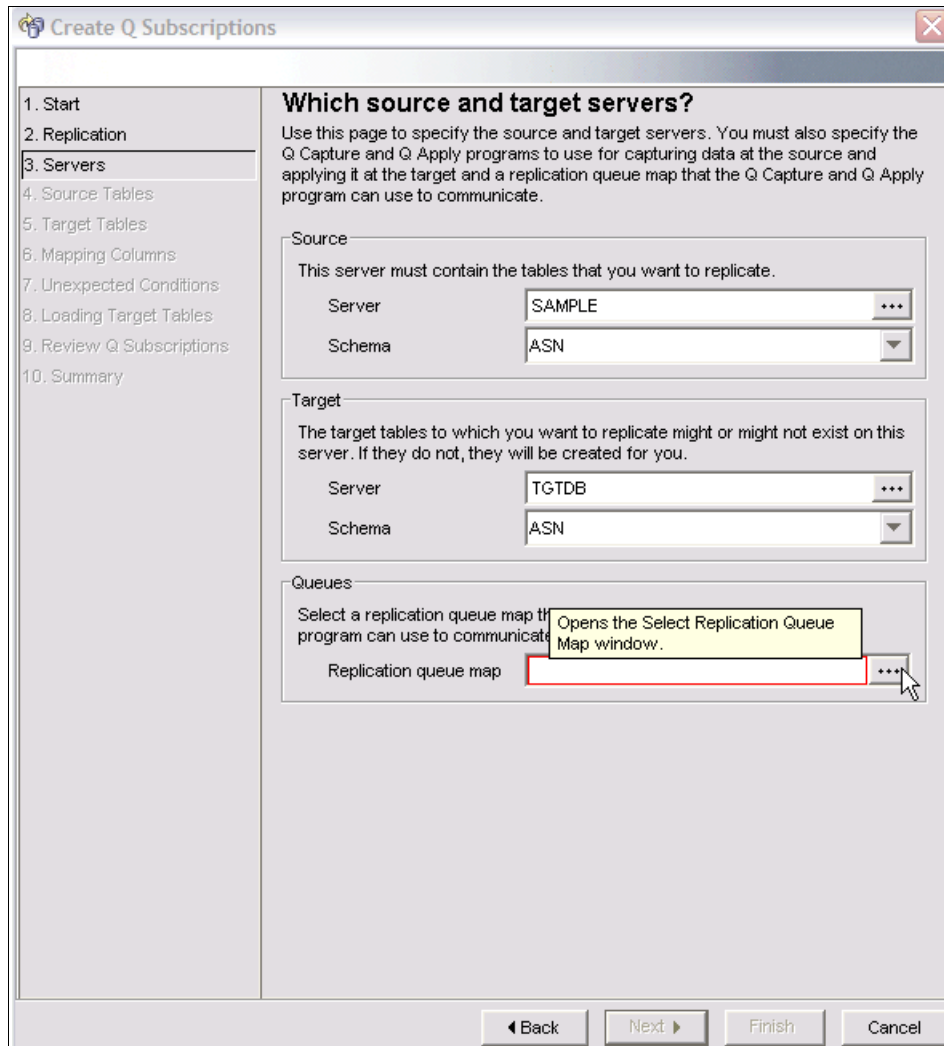


Figure 13-15 Create Q Subscriptions Wizard Servers before

- e. Click **New** on the Select Replication Queue Map window. See Figure 13-16.

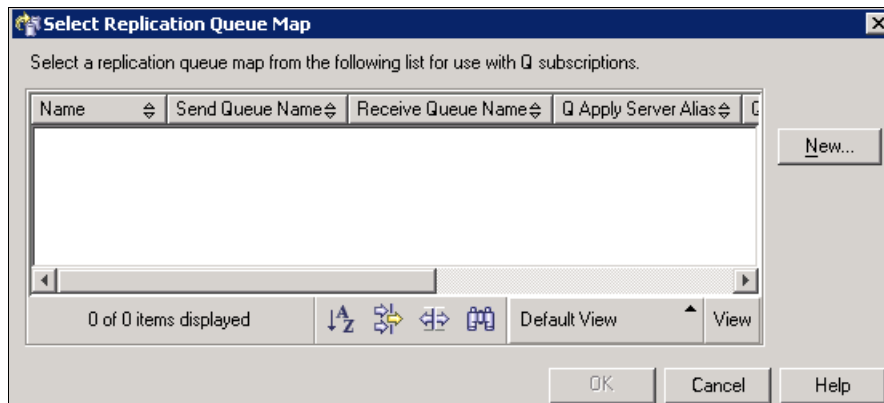


Figure 13-16 Select Replication Queue Map New

- f. In the Create Replication Queue Map window under the General tab, insert the Send queue, Receive queue, and Administration queue. See Figure 13-17.

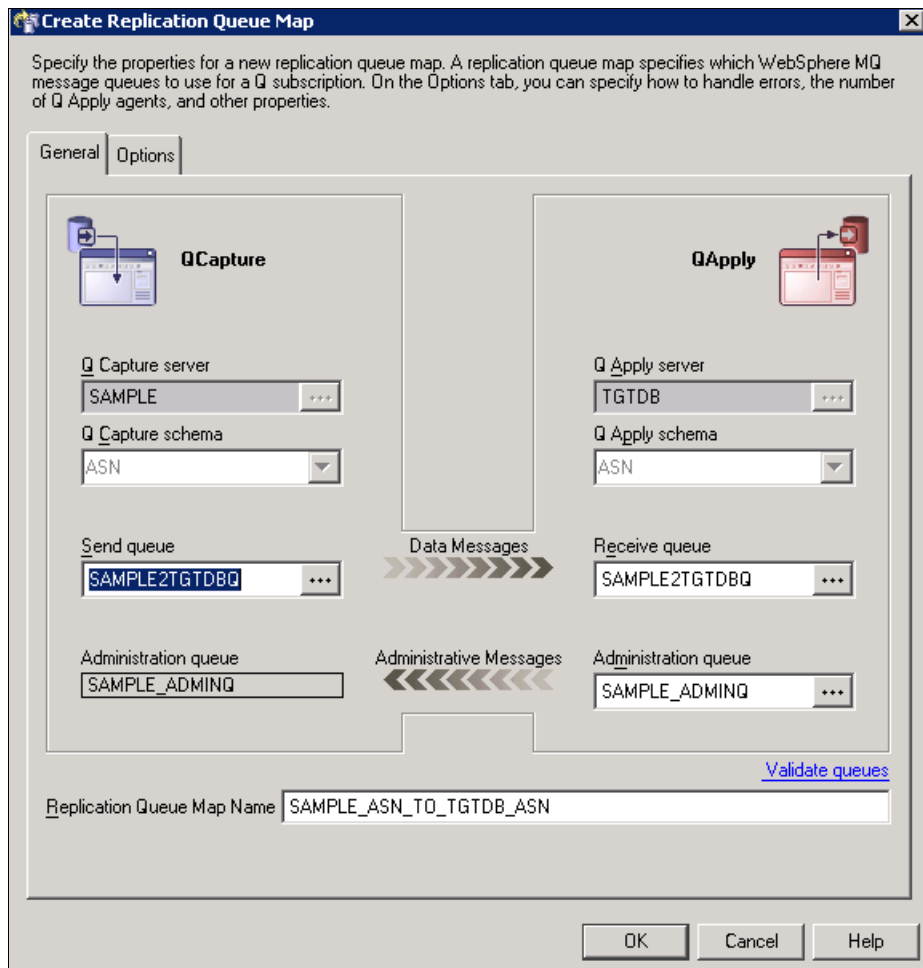


Figure 13-17 Create Replication Queue Map General tab



- g. In the Create Replication Queue Map window under the Options tab, you can specify the attributes for the new replication queue map. For our example, we kept the default values. Click **OK** to finish the new replication queue map. See Figure 13-18.

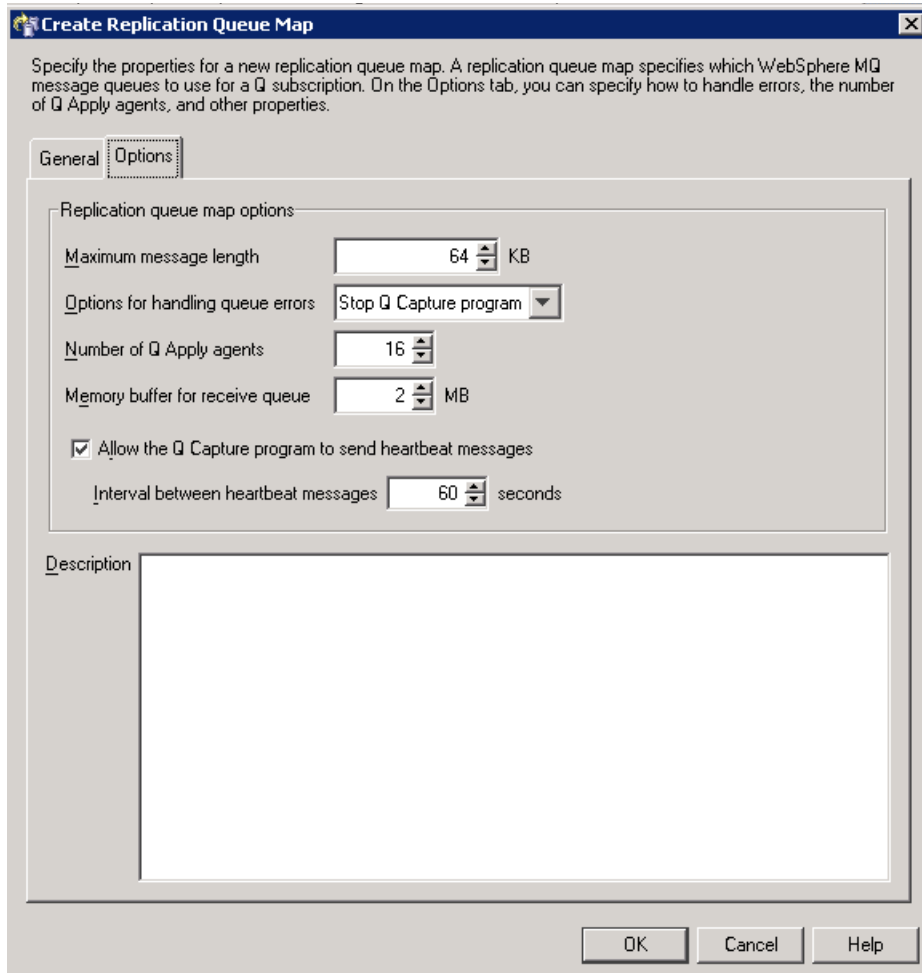


Figure 13-18 Create Replication Queue Map Options tab

- h. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 13-19. After saving the SQL, select **Run now** and click **OK**.

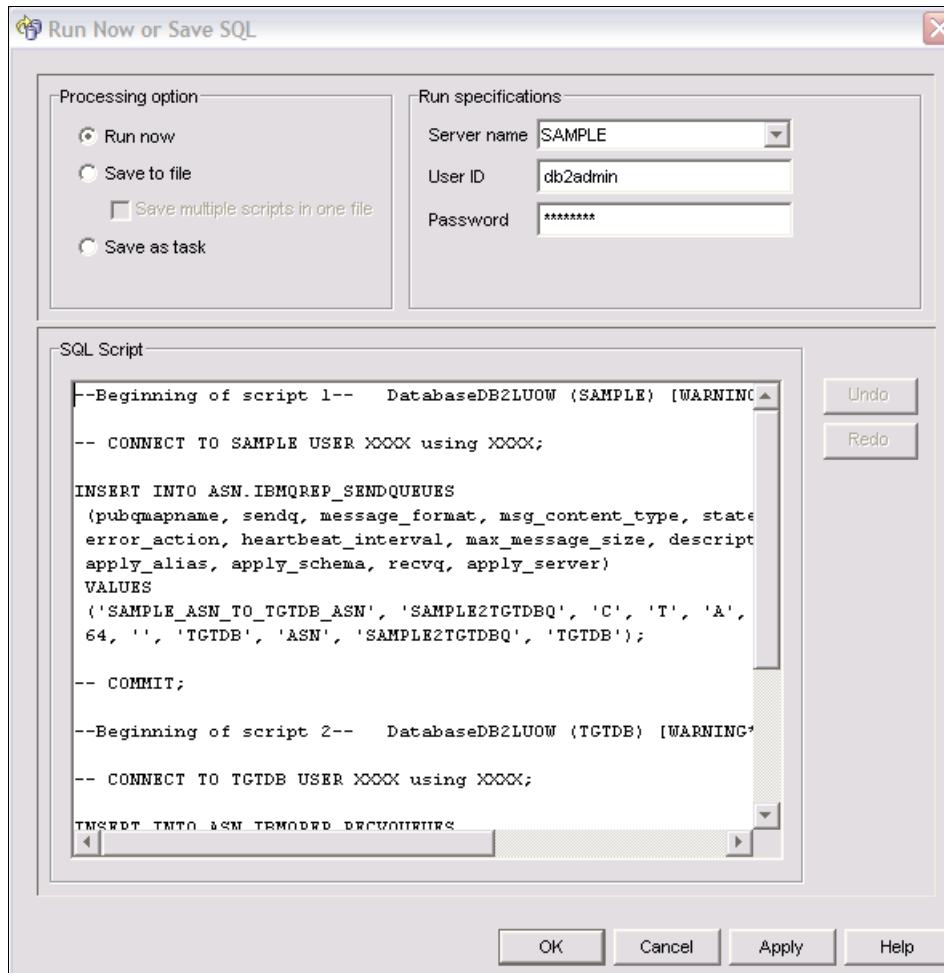


Figure 13-19 Run Now or Save SQL Queue Map

- i. From the Select Replication Queue Map, highlight your newly created queue map, then click **OK**. See Figure 13-20.

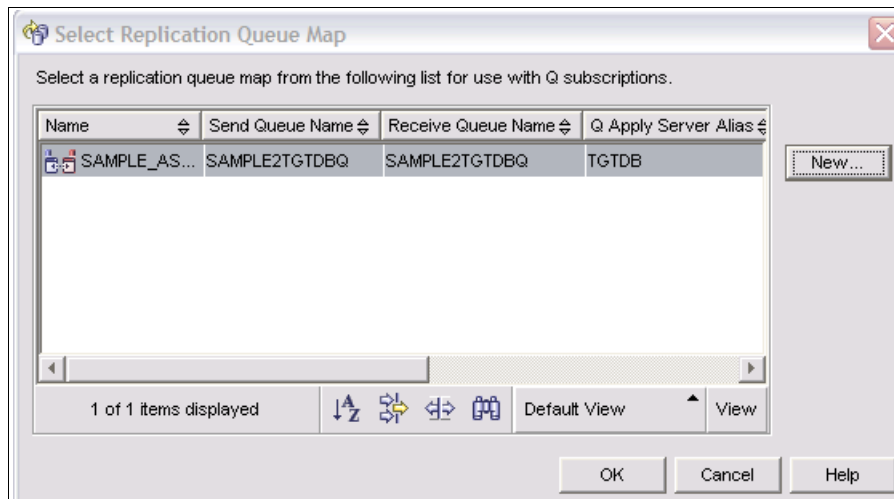


Figure 13-20 Select Replication Queue Map selected

- j. Now you should be back to the Create Q Subscription wizard panel with all of the information filled in for 3. Servers. See Figure 13-21. Click **Next** to continue.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target Tables  
6. Mapping Columns  
7. Unexpected Conditions  
8. Loading Target Tables  
9. Review Q Subscriptions  
10. Summary

### Which source and target servers?

Use this page to specify the source and target servers. You must also specify the Q Capture and Q Apply programs to use for capturing data at the source and applying it at the target and a replication queue map that the Q Capture and Q Apply program can use to communicate.

**Source**  
This server must contain the tables that you want to replicate.

Server: SAMPLE  
Schema: ASN

**Target**  
The target tables to which you want to replicate might or might not exist on this server. If they do not, they will be created for you.

Server: TGTDB  
Schema: ASN

**Queues**  
Select a replication queue map that the Q Capture program and Q Apply program can use to communicate.

Replication queue map: SAMPLE\_ASN\_TO\_TGTDB\_ASN

◀ Back   Next ▶   Finish   Cancel

Figure 13-21 Create Q Subscriptions panel 3 complete

- k. Select the tables from the source you wish to replicate. In our example, we chose one table to replicate. See Figure 13-22. Click **Next** to continue.

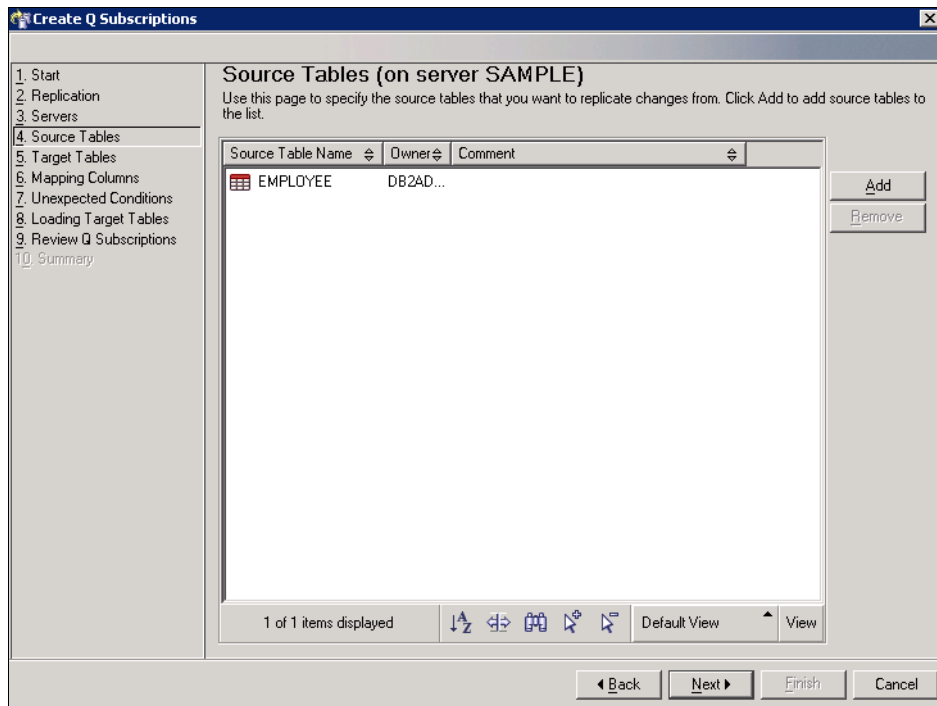


Figure 13-22 Create Q Subscriptions source tables

- I. For step 5. Target, in the Create Q Subscriptions wizard, select the target type you want. For our example, we chose to create a new table on the target. See Figure 13-23. Click **Next** to continue.

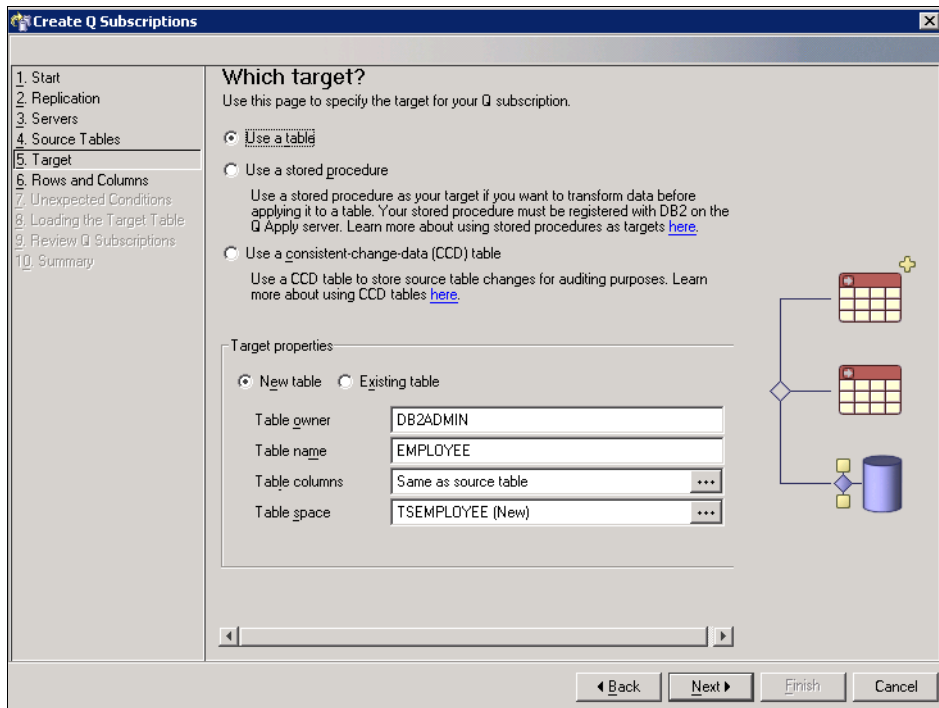


Figure 13-23 Create Q Subscriptions target tables

- m. For step 6. Rows and Columns, in the Create Q Subscriptions wizard, you have the choice of granularity with your columns and rows replicated. See Figure 13-24. Click **Next** to continue.

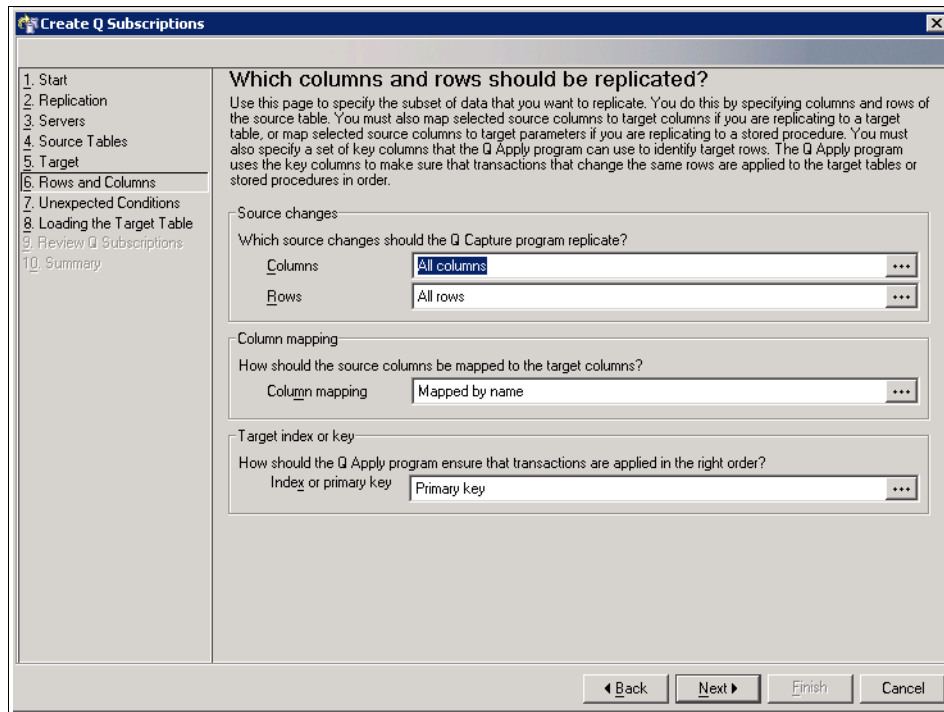


Figure 13-24 Create Q Subscriptions replicated columns and rows

- n. For step 7. Unexpected Conditions, in the Create Q Subscriptions wizard, you can specify how you want the Q Apply program to handle unexpected conditions. See Figure 13-25. Click **Next** to continue.

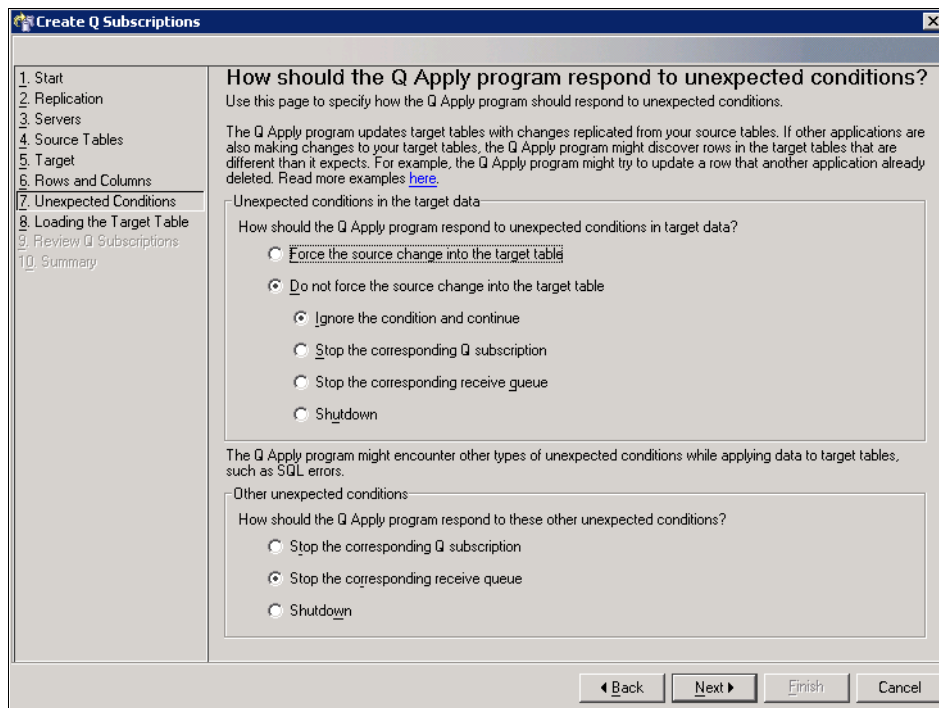


Figure 13-25 Create Q Subscriptions Unexpected Conditions



- o. For step 8. Loading the Target Table, specify how you want your target tables loaded. See Figure 13-26. Click **Next** to continue.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target  
6. Rows and Columns  
7. Unexpected Conditions  
**8. Loading the Target Table**  
9. Review Q Subscriptions  
10. Summary

### How should the target table be loaded?

Use this page to specify if and how your target table will be loaded. When the Q subscription starts, the Q Apply program can initiate a load of the target table with the utility or utilities that you specify.

Loading the target table

How will the target table be loaded when this Q subscription starts?

- Automatic: The Q Apply program performs the load
  - How should the Q Apply program choose which load method to use?
    - Best available: Let the Q Apply program choose a load method
    - Always use LOAD FROM CURSOR
    - Always use export and load
    - Always use export and import
  - Manual: You perform the load manually and inform the Q Apply program when the load is complete
  - None: The target table will not be loaded

Nickname

Which nickname should Q Apply use to load rows from the source table?

- Create a new nickname
  - Nickname owner:
  - Nickname name:
  - Server definition:
- Use an existing nickname

◀ Back   Next ▶   Finish   Cancel

Figure 13-26 Create Q Subscription Loading the Target Table

- p. For step 9. Review Q Subscriptions, in the Create Q Subscriptions wizard, observe the settings. See Figure 13-27. Click **Next** to continue.

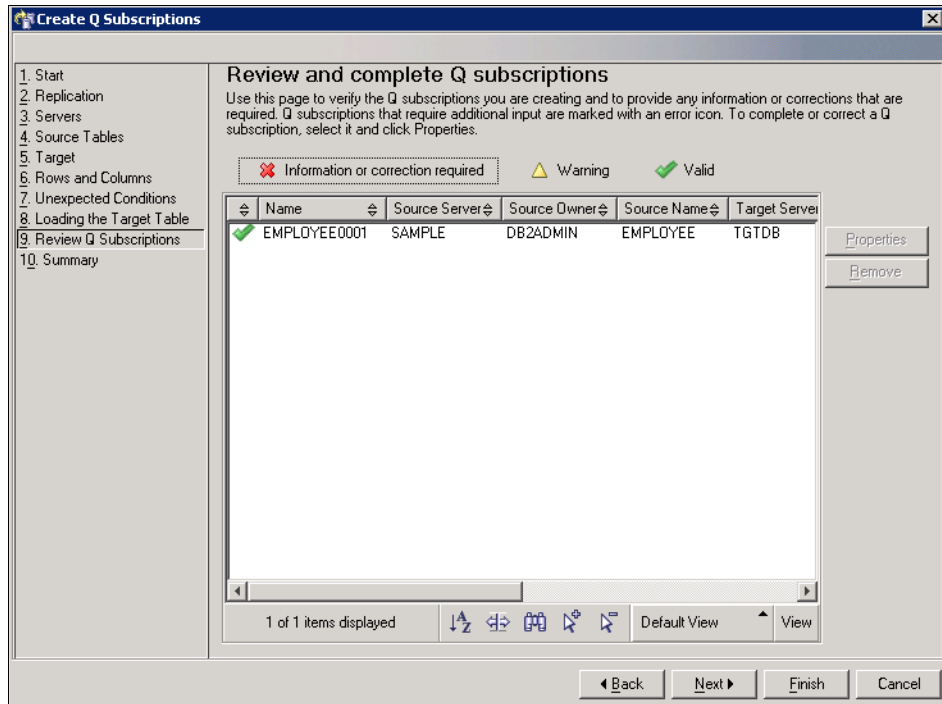


Figure 13-27 Create Q Subscriptions, Review Q Subscriptions

- q. Step 10. Summary, in the Create Q Subscriptions wizard, is the final window before creating the SQL. See Figure 13-28. Click **Finish** to create the SQL.

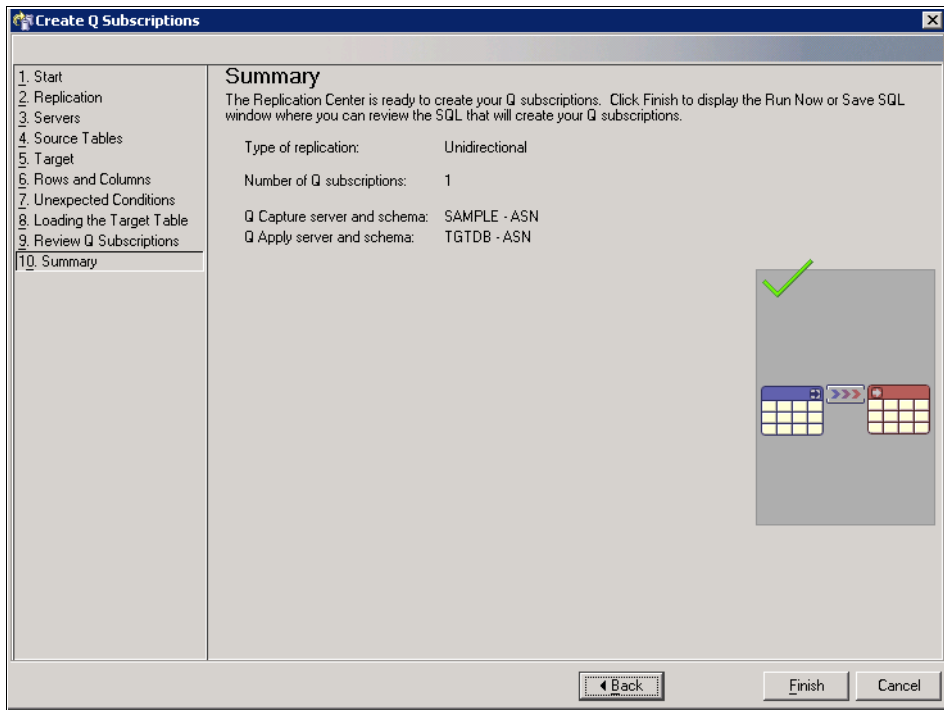


Figure 13-28 Create Q Subscription Summary

- r. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 13-29. After saving the SQL, select **Run now** and click **OK**.

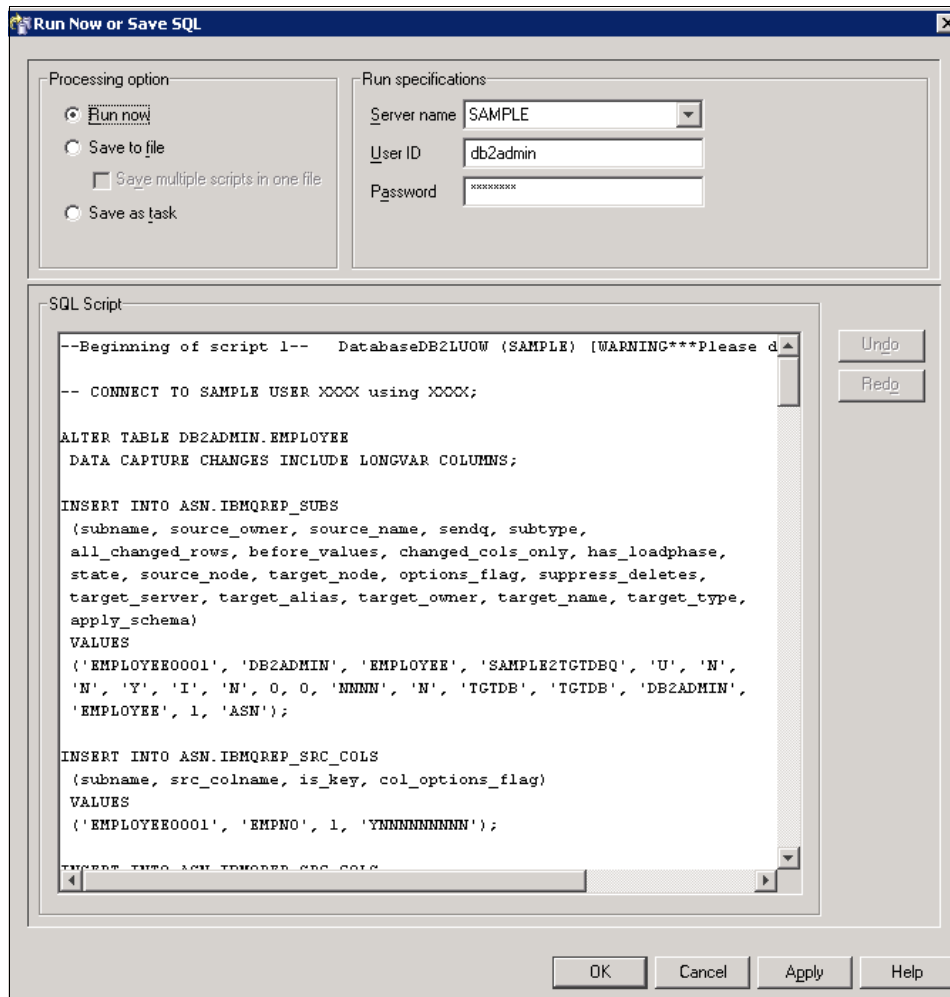


Figure 13-29 Run Now or Save SQL Q Subscriptions

9. Configure target to source server connectivity.

If you plan to have the Q Apply program automatically load targets with source data by using the EXPORT utility, as we did in our example, the Q Apply program must be able to connect to the Q Capture server. This connection requires a password file that is created with the **asnpwd** command.

- a. Create a directory on the Q Apply Control server where you plan to run the Q Apply start command. In our example, we used `c:\replwork\`.
- b. Run the command to create the password file in the directory that you have created. See Example 13-5.

```
asnpwd init
```

*Example 13-5 asnpwd init*

---

```
C:\replwork> asnpwd init
2006-10-25-16.31.47.671000 ASN1981I "Asnpwd" : "" : "Initial". The
program completed successfully using password file "asnpwd.aut".
```

---

- c. Run the command to update the `asnpwd.aut` file with the connection information. See Example 13-6.

```
asnpwd add alias SAMPLE id db2admin password adminpw
```

*Example 13-6 asnpwd add*

---

```
C:\replwork> asnpwd add alias SAMPLE id db2admin password adminpw
2006-10-25-16.37.10.535000 ASN1981I "Asnpwd" : "" : "Initial". The
program completed successfully using password file "asnpwd.aut".
```

---

The unidirectional configuration has been completed. We can start the Q replication process. When starting the Q replication, start Q Capture first, then Q Apply. To stop the replication process, stop Q Apply, then Q Capture.

## Start Q Capture

This can be done either from the command line or from the Replication Center.

### **Command line**

To start Q Capture from the command line, change to the desired directory and issue the **start** command. See Example 13-7.

```
asncap capture_server=SAMPLE
```

#### *Example 13-7 asncap*

---

```
C:\replwork> asncap capture_server=SAMPLE  
2006-10-25-16.49.51.309000 ASN7000I "Q Capture" : "ASN" :  
"WorkerThread" : "1" subscriptions are active. "0" subscriptions are  
inactive. "0" subscriptions that were new and were successfully  
activated. "0" subscriptions that were new could not be activated and  
are now inactive.  
2006-10-25-16.49.51.329000 ASN0572I "Q Capture" : "ASN" :  
"WorkerThread" : The program initialized successfully.
```

---

## Replication Center

To start Q Capture, complete the following steps:

1. In the Replication Center, expand **Q Replication** → **Operations** → **Q Capture Servers**. In the right pane, right-click the server on which you are starting Q Capture. See Figure 13-30.

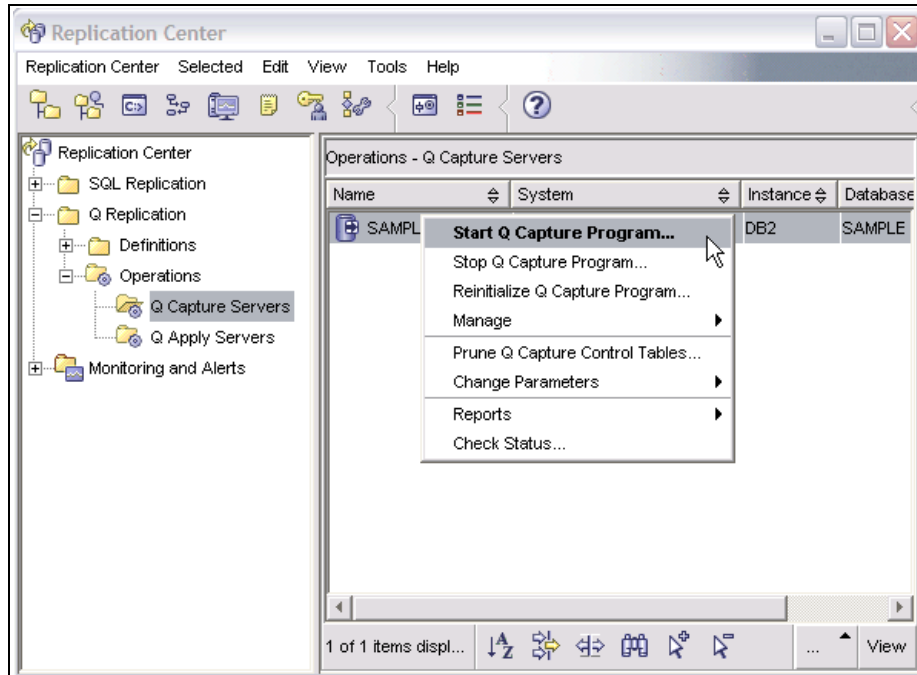


Figure 13-30 Replication Center Start Q Capture

- From the Run Now or Save Command window, insert the required information in to the User ID, Password and start Directory fields. Click **OK** to issue the command. See Figure 13-31.

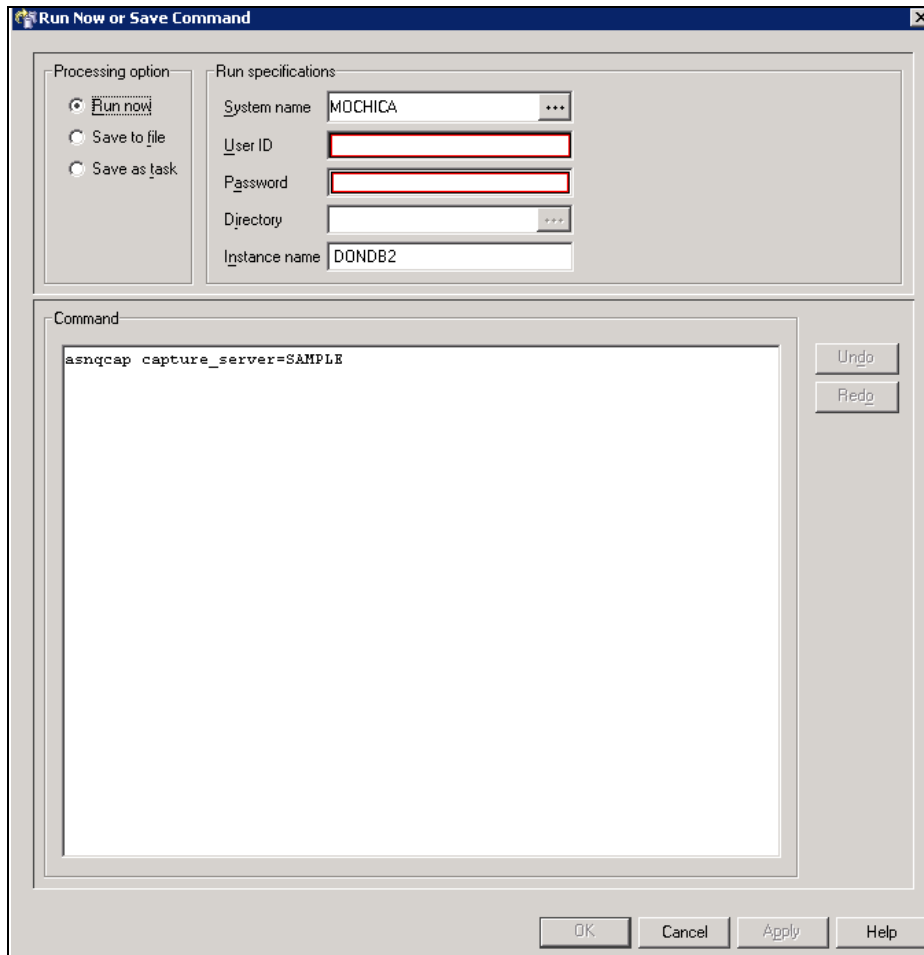


Figure 13-31 Run Now or Save Command Q Capture Start

## Start Q Apply

This can be done from either the command line or from the Replication Center.



## Command line

To start Q Apply from the command line, change to the desired directory and issue the start command. See Example 13-8.

```
asnaapp apply_server=TGTDDB
```

### Example 13-8 asnaapp

```
C:\replwork> asnaapp apply_server=TGTDDB
2006-10-25-15.58.20.785000 ASN7526I "Q Apply" : "ASN" : "BR00000" :
The Q Apply program has started processing the receive queue
"SAMPLE2TGTDDBQ" for replication queue map "SAMPLE_ASN_TO_TGTDDB_ASN".
```

## Replication Center

To start Q Apply, use the following steps:

1. In the Replication Center, expand **Q Replication** → **Operations** → **Q Apply Servers**. In the right pane, right-click the server on which you are starting Q Apply. See Figure 13-32.

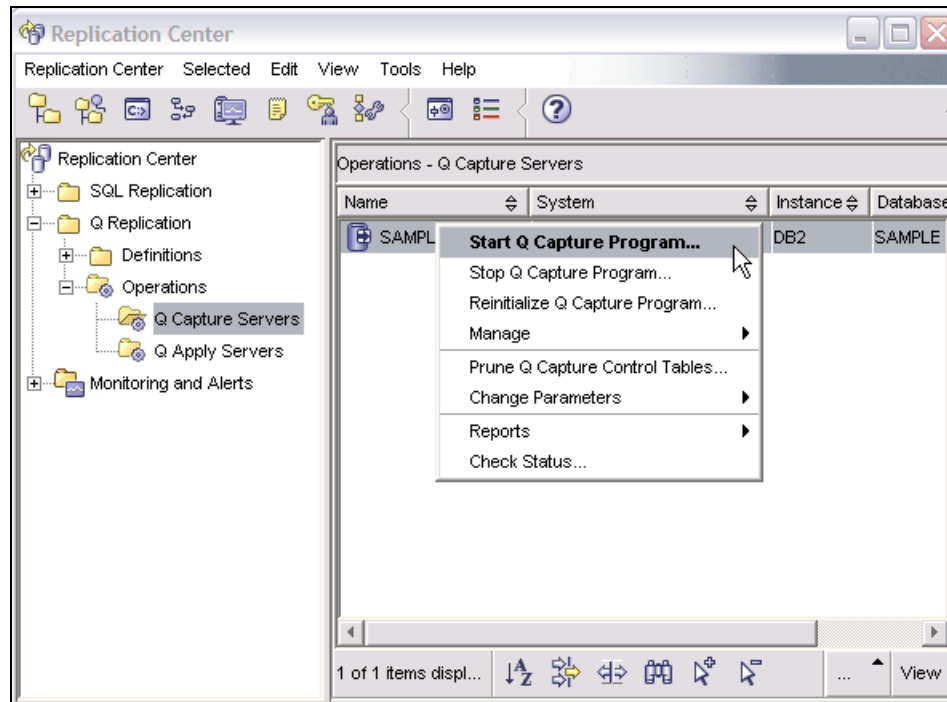


Figure 13-32 Replication Center Start Q Apply

- From the Run Now or Save Command window, insert the required information into the User ID, Password and start Directory fields. Click **OK** to issue the command. See Figure 13-33.

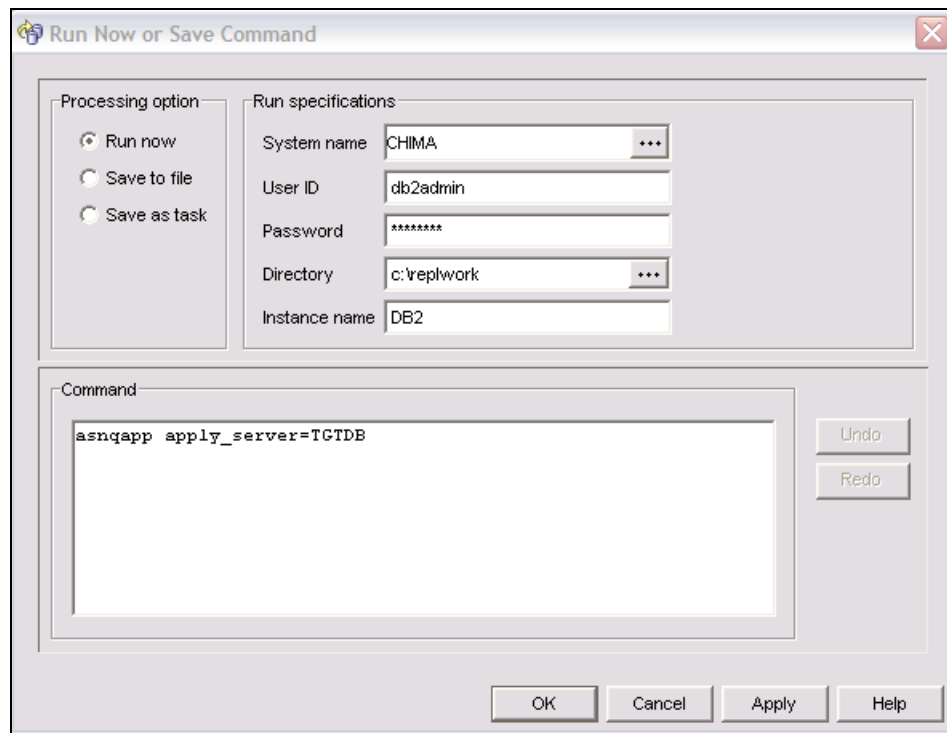


Figure 13-33 Run Now or Save Command Start Q Apply



## Backup and Recovery

In this chapter, we cover the new versatile and ease-of-use Backup and Recovery features provided in DB2 V9.1 and V9.5.

Single System View (SSV) greatly simplifies backing up multiple-partition databases. SSV also makes the process of recovery to a single stable point in time more robust.

The snapshot backup, integrated in DB2 9.5, provides a solution for many sites which require either 24x7 uptime, or which have such large databases that a conventional backup is simply not practical.

While DB2 has provided a complete solution for recovery to point in time or to end of logs, the **restore database** and **rollforward database** commands have kept the process of recovery in two discrete steps. DB2 V9 changed this with the **recover database** command, and DB2 V9.5 has added a valuable feature to the **rollforward database** command.

Throughout the progression of DB2 versions on Linux, UNIX, and Windows platforms, there have been incremental improvements in how administrators can manage backups and archived transaction log files. We examine how DB2 9.5 provides a major milestone in automatic management of these files, integrated into the API and the command line.

## 14.1 Single system view backup

The single system view (SSV) backup is a new function introduced in DB2 V.9.5. SSV provides the capability to perform the multi-partitioned database backup like a single system.

Prior to DB2 9.5, to take the database backup for a multi-partitioned database, you either run the backup command on each partition one at a time or use the **db2\_a11** command to run the backup commands in parallel. However, under the covers, the database backup takes place on each node independently even with the **db2\_a11** command. The result is that the backup image files have different backup timestamps regardless of how the database backup commands are run. When a database restore is required, identifying all database partition backup images that have different backup timestamps for the same backup becomes complicated. Besides, the log files required for point-in-time recovery cannot be included in the backup image. Determining the minimum recovery time for the backup that contains all those database partitions is also difficult.

These difficulties have been removed on DB2 V9.5 by the single system view backup. When performing a backup operation from the catalog node of a multi-partitioned database, you can specify one, a few, or all partitions to be included in the backup. The specified partitions are backed up simultaneously, and the backup timestamp associated with all specified database partitions is the same. In addition, you can include database logs with a SSV backup. When restoring the database from a SSV backup image, one backup image can be used for all partitions and the required log files are included in the backup image for point-in-time recovery.

### 14.1.1 Using SSV

There is no installation or configuration required for using the SSV backup. The SSV backup is enabled when you specify the keyword **ON DBPARTITIONNUM** or **ON ALL DBPARTITIONNUMS** in the database backup command.

You can use the “**INCLUDE LOGS**” option to include the logs in the backup image for a multi-partitioned database backup. SSV backup packages all the log files required to recover the database to a point-in-time.

#### Performing the backup task

To take a database backup, you can run the backup command as usual on the catalog partition. These are two simple SSV backup command samples:

```
BACKUP DB sample ON ALL DBPARTITIONNUMS TO <backup directory>
BACKUP DB sample ON DBPARTITIONNUMS 0 TO 3 TO <backup directory>
```

The first one specifies the keyword ON ALL DBPARTITIONNUMS telling DB2 to back up all database partitions listed in the db2nodes.cfg file. The second one specifies the keyword ON DBPARTITIONNUMS 0 TO 3 instructing DB2 to back up partitions with the database partition number 0 to 3.

When you want to include the log files for minimum recovery, all you have to do is to set the keyword ONLINE as in this sample command:

```
BACKUP DB sample ON ALL DBPARTITIONNUMS ONLINE TO <backup directory>
```

In DB2 9.5, specifying the ONLINE keyword in the backup command activates the INCLUDE LOGS option.

Example 14-1 shows that the database backup of the multi-partitioned database SAMPLE is taken with a single command and all four database partitions have the same timestamps.

*Example 14-1 The sample of the SSV backup*

---

```
$ db2 "backup db sample ON ALL DBPARTITIONNUMS online to /work/backup"
```

```
Part Result
```

```
-----  
0000 DB20000I The BACKUP DATABASE command completed successfully.  
0001 DB20000I The BACKUP DATABASE command completed successfully.  
0002 DB20000I The BACKUP DATABASE command completed successfully.  
0003 DB20000I The BACKUP DATABASE command completed successfully.
```

Backup successful. The timestamp for this backup image is : 20080508160439

```
$ ls /work/backup
```

```
total 681816
```

```
SAMPLE.0.db2inst3.NODE0000.CATN0000.20080508160439.001
```

```
SAMPLE.0.db2inst3.NODE0001.CATN0000.20080508160439.001
```

```
SAMPLE.0.db2inst3.NODE0002.CATN0000.20080508160439.001
```

```
SAMPLE.0.db2inst3.NODE0003.CATN0000.20080508160439.001
```

---

### **Checking the log file status**

You can check if one database backup image includes the log files using the **db2ckbkp** command. This command provides the information about the specified database backup image file.

```
db2ckbkp -h <database backup file>
```

Example 14-2 shows a sample output of the **db2ckbkp -h** command. The line field, Includes Logs, tells if the backup image includes the log files or not. The value **1** means that this backup includes the log files.

### Example 14-2 Sample output of the db2ckbcp command

---

```
$ db2ckbcp -h SAMPLE.0.db2inst3.NODE0001.CATN0000.20080506162459.001
```

```
=====
MEDIA HEADER REACHED:
=====
Server Database Name          -- SAMPLE
  Server Database Alias      -- SAMPLE
  Client Database Alias      -- SAMPLE
  Timestamp                  -- 20080508160439
  Database Partition Number  -- 0
  Instance                   -- db2inst3
  Sequence Number            -- 1
  Release ID                  -- C00
  Database Seed               -- 88661681
  DB Comment's Codepage (Volume) -- 0
  DB Comment (Volume)        --
  DB Comment's Codepage (System) -- 0
  DB Comment (System)        --
  Authentication Value        -- 255
  Backup Mode                 -- 1
  Includes Logs              -- 1
  Compression                 -- 0
  Backup Type                 -- 0
  Backup Gran.                -- 0
  Status Flags                -- 11
  System Cats inc             -- 1
  Catalog Partition Number   -- 0
  DB Codeset                  -- UTF-8
  DB Territory                --
  LogID                       -- 1210221761
  LogPath                     --
/home/db2inst3/db2/db2inst3/NODE0000/SQL00001/SQLLOGDIR/
  Backup Buffer Size          -- 3280896
  Number of Sessions         -- 1
  Platform                   -- 14...
```

---

### Restoring the database and the LOGTARGET keyword

In DB2 9.5, the online backup command automatically takes the needed log files into the backup image for minimum forward recovery. However, to use the log files for rollforward recovery, you have to extract these files out of the database backup image. It is not done automatically. There are two ways to do this:

- ▶ Specify the LOGTARGET keyword in the restore database command. DB2 extracts the log files included in the backup image to the target directory specified by the LOGTARGET during the database restore process. The directory specified in the LOGTARGET must be an existing directory.

Each partition should have its own directory or subdirectory because the log file names can be the same on each database partition:

```
RESTORE DB <DB name> FROM <backup dir> TAKEN AT <backup timestamp>  
ON <target db path> LOGTARGET <target log restore path>
```

- Specify the LOGS keyword in the restore database command. When you specify this keyword, the database restore is not executed. This command only takes the log files out of the backup image to the directory specified in the LOGTARGET keywords.

The target directory must have been created already.

```
RESTORE DB <DB name> LOGS FROM <backup dir> TAKEN AT <backup  
timestamp> ON <target db path> LOGTARGET <target log restore path>
```

Example 14-3 shows one way to restore a four partition database. The backup image was taken using SSV backup as shown in Example 14-1 on page 715. In this restore sample, we run the restore command with the LOGTARGET keyword on each partition to restore one partition at a time. Note that same database backup image is used in each restore. The directory for holding the log files is created manually in each physical node before restore starts.

*Example 14-3 The sample of restore database with the LOGTARGET keyword*

---

```
$ export DB2NODE=0  
$ db2 terminate  
DB20000I The TERMINATE command completed successfully.  
$ db2 "restore db sample from /work/backup taken at 20080508160439 on  
/home/tukiv95/db2 logtarget /work/logs/log0"  
DB20000I The RESTORE DATABASE command completed successfully.  
$ export DB2NODE=1  
$ db2 terminate  
DB20000I The TERMINATE command completed successfully.  
$ db2 "restore db sample from /work/backup taken at 20080508160439 logtarget  
/work/logs/log1 without prompting"  
SQL2540W Restore is successful, however a warning "2523" was encountered  
during Database Restore while processing in No Interrupt mode.  
$ export DB2NODE=2  
$ db2 terminate  
DB20000I The TERMINATE command completed successfully.  
$ db2 "restore db sample from /work/backup taken at 20080508160439 logtarget  
/work/logs/log2 without prompting"  
SQL2540W Restore is successful, however a warning "2523" was encountered  
during Database Restore while processing in No Interrupt mode.  
$ export DB2NODE=3  
$ db2 terminate  
DB20000I The TERMINATE command completed successfully.  
$ db2 "restore db sample from /work/backup taken at 20080508160439 logtarget  
/work/logs/log3 without prompting"
```

SQL2540W Restore is successful, however a warning "2523" was encountered during Database Restore while processing in No Interrupt mode.

```
$ ls -l /work/logs/log*
```

```
/work/logs/log0:
```

```
total 24
```

```
-rw----- 1 tukiv95 system      12288 May  8 17:47 S0000005.LOG
```

```
/work/logs/log1:
```

```
total 24
```

```
-rw----- 1 tukiv95 system      12288 May  8 17:47 S0000005.LOG
```

```
/work/logs/log2:
```

```
total 24
```

```
-rw----- 1 tukiv95 system      12288 May  8 17:47 S0000005.LOG
```

```
/work/logs/log3:
```

```
total 24
```

```
-rw----- 1 tukiv95 system      12288 May  8 17:48 S0000005.LOG
```

---

**Tips:** You also can use **db2\_a11** to run the **restore** command for all database partitions. The command syntax is as follows:

► For the **restore** command for the catalog partition:

```
$ db2_a11 "\<<+0< db2 restore db sample from /work/backup taken  
at 20080508160439 on /home/tukiv95/db2 logtarget /work/logs/log##  
without prompting"
```

► For the restore command for the other partitions:

```
$ db2_a11 "\||<<-0< db2 restore db sample from /work/backup  
taken at 20080508160439 logtarget /work/logs/log## without  
prompting"
```

The first command is the **restore** command for the catalog partition, and the second one is for all other non-catalog database partitions. You have to restore the catalog partition first. After that, you can run the **restore** command for non-catalog database partitions in parallel. The keyword **||<<-0<** in front of the second command means that the **db2\_a11** command executes this statement in parallel on all partitions except for partition number zero, the catalog partition.

The **##** is part of the target directory. The **\"** keyword (a back slash and a double quote) in front of the command tells DB2 to replace the keyword **##** with the database partition number under which the restore command is run.



## Rollforward to the minimum recovery time

Prior to DB2 9.5, because the backup timestamps are different in each backup image, to do the forward recovery, you have to identify the minimum (latest) recovery timestamp from the database backup images. In DB2 9.5, SSV backup has one backup timestamp for all backup images.

All you have to do is to specify the backup timestamp in the keyword, TO END OF BACKUP, in the **rollforward** command:

```
ROLLFORWARD DATABASE <database name> TO END OF BACKUP ON ALL
DBPARTITIONNUMS AND COMPLETE OVERFLOW LOG PATH ( <general overflow
log path> , <overflow log path for each db partition> , ... )
```

The **rollforward** database command tries to search the log files from the log path, the archive log path, and the failover log path by default. If the log files are in the directory other than these directories, specify the log file location in the OVERFLOW LOG PATH. For the partitioned database environment, the *general overflow log path* is the default overflow log path for all database partitions. You need to specify one even if you do not use it. The overflow log paths for each database partition are listed after this path.

Example 14-4 is the sample for the END OF BACKUP rollforward recovery in a partitioned database environment. In this example, we specify four directories as the overflow log path for each database partition.

*Example 14-4 Sample of rollforward recovery with the END OF BACKUP keyword*

---

```
$ export DB2NODE=0
$ db2 terminate
DB20000I The TERMINATE command completed successfully.
$ db2 "rollforward database sample"
```

### Rollforward Status

```
Input database alias           = sample
Number of nodes have returned status = 4
```

Nodenumbr	Rollforward status	Next log to be read	Log files processed	Last committed transaction
0	DB pending	S0000005.LOG	-	2008-05-08-07.04.53.000000 UTC
1	DB pending	S0000005.LOG	-	2008-05-08-07.04.51.000000 UTC
2	DB pending	S0000005.LOG	-	2008-05-08-07.04.51.000000 UTC
3	DB pending	S0000005.LOG	-	2008-05-08-07.04.52.000000 UTC

```
$ db2 "ROLLFORWARD DATABASE SAMPLE TO END OF BACKUP ON ALL DBPARTITIONNUMS AND
COMPLETE
> OVERFLOW LOG PATH
> ( /work/logs,
> /work/logs/log0 on DBPARTITIONNUM 0,
```

```
> /work/logs/log1 on DBPARTITIONNUM 1,
> /work/logs/log2 on DBPARTITIONNUM 2,
> /work/logs/log3 on DBPARTITIONNUM 3)"
```

#### Rollforward Status

```
Input database alias           = sample
Number of nodes have returned status = 4
```

Node number	Rollforward status	Next log to be read	Log files processed	Last committed transaction
0	not pending		S0000005.LOG-S0000006.LOG	2008-05-08-07.04.53.000000 UTC
1	not pending		S0000005.LOG-S0000006.LOG	2008-05-08-07.04.51.000000 UTC
2	not pending		S0000005.LOG-S0000006.LOG	2008-05-08-07.04.51.000000 UTC
3	not pending		S0000005.LOG-S0000006.LOG	2008-05-08-07.04.52.000000 UTC

```
DB20000I The ROLLFORWARD command completed successfully.
```

## 14.1.2 Considerations

Here are a few considerations for using the SSV backup:

- ▶ The SSV backup command must be executed on the catalog partition.  
If you execute the SSV backup command on the non catalog partition, the backup command might fail with an SQL4976N message.
- ▶ The SSV backup is not enabled automatically.  
Without specifying the keyword ON DBPARTITIONNUM or ON ALL DBPARTITIONNUMS in the backup command line, the backup command is executed on the current database partition only.
- ▶ Only one backup task can be executed in parallel on the database partition

You must not execute the multiple database backup on one database partition. If you execute the backup command on the non-catalog partition during the SSV backup running, additional backup commands might fail with an SQL1035N message.

## 14.2 Backup and restore database with snapshot backup

Many storage subsystems provide FlashCopy or snapshot functions, which allow you to make nearly instantaneous point in time copies of entire logical volumes or data sets.

In a traditional backup or restore operation, the database manager copies data to or from disks using operating system calls. Being able to use the storage subsystems to perform the data copying makes the backup and restore operations much faster. In addition, the impact to the ongoing operation is minimum, therefore, it is an optimal solution for a 24X7 production system.

Prior to DB2 9.5, the FlashCopy backup and restore support from DB2 is a manual process. The procedure to do FlashCopy backup in DB2 9 is as follows:

1. Identify Logical Unit Numbers (LUNs) associated with the database as FlashCopy source.
2. Identify free LUNs as FlashCopy target.
3. Establish the FlashCopy pairs from a storage management interface.
4. Issue the **set write suspend for database** command to tell DB2 to suspend write I/Os.
5. Issue the storage commands to do the actual FlashCopy.
6. Issue the **set write resume for database** command to resume write I/Os.

If you want to restore database from FlashCopy backup, do the following steps:

1. Restore or copy target LUNs containing backup images.
2. Issue the **db2inidb** command to initialize the database for rollforward recovery.
3. Issue the DB2 **rollforward** command to roll forward database logs and make the database available to use.

Because DB2 9.5, DB2 provides integrated support for snapshot backups through DB2 Advanced Copy Services (ACS). DB2 ACS enables you to use the fast copying technology of a storage device to perform the data copying part of backup and restore operations. Table 14-3 on page 728 shows storage systems supported by DB2 ACS.

A backup operation that uses DB2 ACS is called a snapshot backup. To do a snapshot backup and restore with DB2 ACS, you only need to specify **use snapshot** in the **backup** and **restore** command.

The default behavior for a snapshot backup is a full database offline backup of all paths that make up the database including all containers, database path (DBPATH), primary log, and mirror log paths (INCLUDE LOGS is the default for all snapshot backups unless EXCLUDE LOGS is explicitly stated).

DB2 9.5 does not support the **use snapshot** parameter in the **backup** command with any of the following parameters:

- ▶ **tablespace**
- ▶ **incremental**
- ▶ **with num-buffers buffers**
- ▶ **buffer**
- ▶ **parallelism**
- ▶ **compress**
- ▶ **util\_impact\_priority**
- ▶ **sessions**

## 14.2.1 DB2 ACS overview

DB2 ACS is a limited version of *Tivoli Storage Manager for Advanced Copy Services - Data Protection for Snapshot Devices for DB2 Advanced Copy Services* (TSM ACS). Table 14-1 lists the limitations of DB2 ACS. Additional functionality can be enabled by using a full licensed TSM ACS.

Table 14-1 Differences between DB2 ACS and TSM ACS

Function	DB2 ACS	TSM ACS
Multiple snapshot versions	Limited to 2	No limit imposed by software
Incremental or no copy FlashCopy	No support	Supported (ESS and DS6000, DS8000 only)
LVM mirroring	A snapshot of both mirrors is done	A snapshot from one mirror set is possible (except IBM System Storage N Series)
Integrated snapshot and tape backup	No support	Supported
Backup from secondary host	No support	Supported
Support for Journaled File System (JFS)	No support (JFS2 only)	JFS (requires backup server) and JFS2

DB2 9.5 GA integrates the TSM ACS Version for DB2 9.5 GA. In this book, all content is based on DB2 9.5 FixPak 1, in which DB2 ACS is a limited version of TSM ACS V5.5. For more information about TSM ACS V5.5, refer to the *TSM ACS V5.5 installation and User's Guide*, SC33-8330, available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

DB2 ACS contains the following components:

- ▶ Snapshot Backup Library (**libacsdb2.a** on AIX, **libacsdb2.so** on Linux)

The Snapshot Backup Library (a vendor library in DB2 terms) provides the necessary extensions in the DB2 environment for implementing snapshot-based backup and restore. The library is also used by **db2acsuti1**.

- ▶ Management Agent (**acsdb**)

The Management Agent (**acsdb**) is the coordinator of a single or multiple partition backup operation. It controls the backup flow and mediates between the application and device agents.

**acsdb** also provides access to the snapshot backup repository, which contains information about the valid snapshot backups and their relationships to snapshot-capable storage devices.

There is one Management Agent per DB2 instance. The Management Agent is normally started from `/etc/inittab`. It needs to be started on one machine only. In a partitioned environment, we suggest starting it in the catalog partition.

- ▶ Device Agent for Common Information Model (CIM) Devices (**acsdbcim**)

The Device Agent for CIM Devices (**acsdbcim**) is the component that invokes a snapshot command on a FlashCopy device (currently ESS800, DS6000, DS8000, or SVC) using the CIM interface.

This agent is also used to update the progress and usability information stored in the (local) snapshot backup repository.

**Note:** CIM is a standard from Distributed Management Task Force (DMTF), which provides a common definition of management information for systems, networks, applications, and services, and allows for vendor extensions. For more information about CIM, refer to:

<http://www.dmtf.org/standards/cim/>

▶ Device Agent for N Series Devices (**acsnnas**, **acsnsan**)

The Device Agent for N Series Devices (**acsnnas** is for NAS attachment, **acsnsan** is for SAN attachment) is the component that invokes a snapshot command on a snapshot capable device (such as N Series) using the NetApp ONTAPI™ interface.

This agent is also used to update the progress and usability information stored in the (local) snapshot backup repository.

▶ Snapshot Object Manager (**db2acsuti1**)

This DB2 ACS component allows DBAs to manage snapshot objects, similarly to what DB2 provides with **db2adut1** to manage DB2 objects stored on TSM. The functionality of **db2acsuti1** includes:

- Querying the list of available DB2 snapshot backups
- Deleting DB2-generated snapshot backups
- Monitoring the status of snapshot backups

**db2acsuti1** uses the same DB2 advanced backup services module as the DB2 backup and restore commands.

▶ Disk Mapper script (**acsdm.sh**)

The Disk Mapper (**acsdm.sh**) is a shell script that provides disk mapping information such as of DS device volume serial numbers to AIX `vpaths` and `hdisk`s. This script is called by **acscim** or **acsnsan** in the snapshot backup and restore cases.

▶ Volume Group Takeover script (**acsvg.sh**)

The Volume Group Takeover utility (**acsvg.sh**) is a shell script that is required only in special high-availability scenarios in which customers use enhanced concurrent capable volume groups on their production systems.

In these cases, this script exports and re-imports the volume groups on an HACMP takeover system after a snapshot restore is performed. This is necessary to synchronize the AIX ODM on the production and HACMP takeover systems.

Figure 14-1 shows overall TSM ACS operating environment for ESS 800, DS6000, and DS8000 using CIM. Note that for DB2 ACS, there is no backup server. For more information, refer to “Operating Environment” in Chapter 1 of the *TSM ACS V5.5 Installation and User’s Guide*, SC33-8330, available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

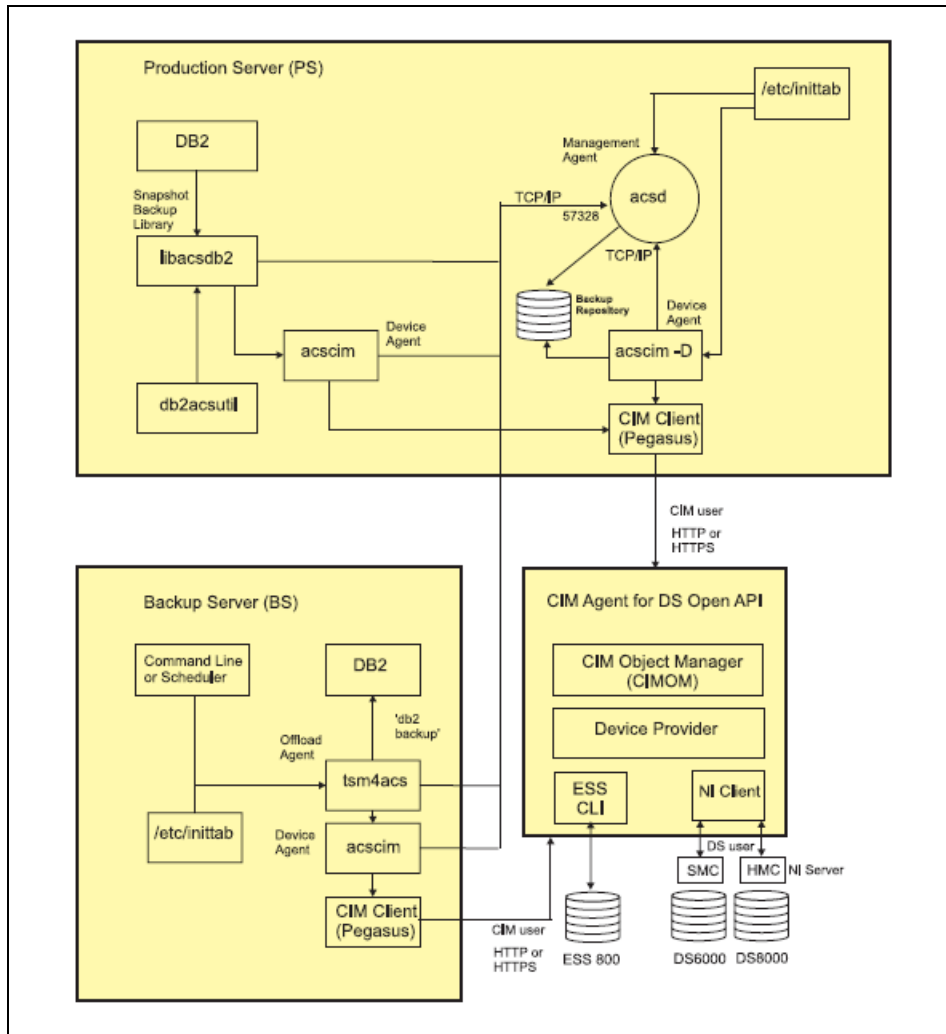


Figure 14-1 TSM ACS Overall operating environment using CIM

## 14.2.2 Storage layout considerations

When designing a database with snapshot backups, you should plan it carefully. The most important factor is the storage layout.

Generally, we suggest the storage layout depicted in Figure 14-2 in a snapshot backup environment. We follow this storage layout in our example, Refer to Figure 14-3 on page 735 and Figure 14-4 on page 749 for details.

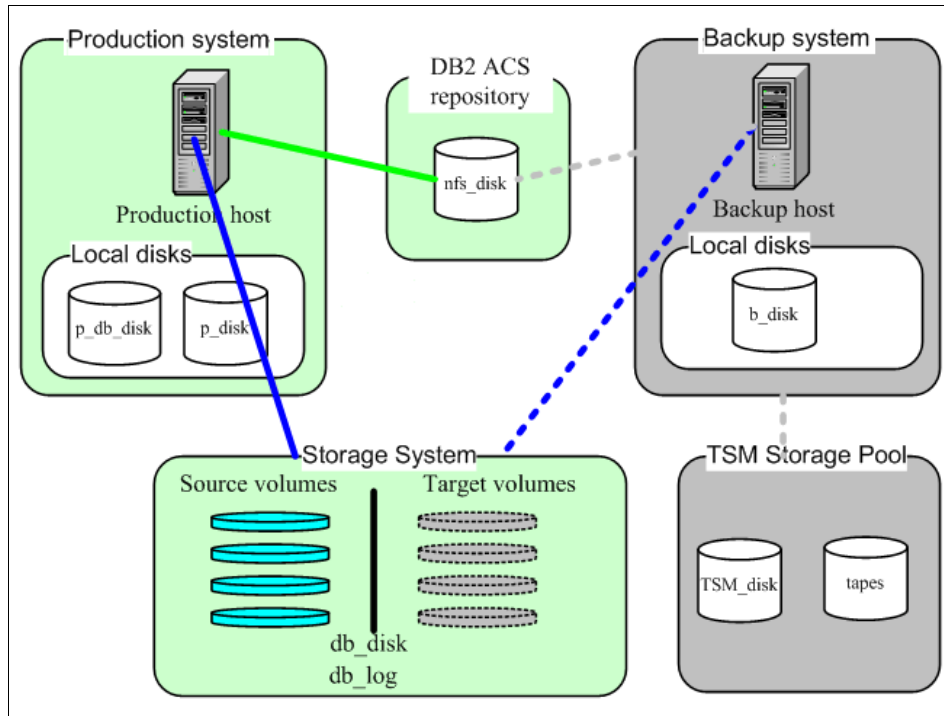


Figure 14-2 Suggested storage layout

We divide the storage to the following categories:

- ▶ **p\_disk:**  
Local disks on the production system are used to install the DB2 product.
- ▶ **Source volumes (disks) on the storage system (db\_data and db\_log categories):**  
The db\_data category contains files such as table space containers and the local database directory, while the db\_log category contains database log files. Make sure that these two categories are on separate volumes. These volumes become the source volumes in the FlashCopy or snapshot operation. Source volumes need to be connected to production hosts.
- ▶ **Target volumes (disks) on the same storage system as source volumes (db\_data and db\_log categories):**  
These volumes become the target volumes in the FlashCopy or snapshot operation. Target volumes need to be connected to backup hosts. Note that if you use DB2 ACS only, there is no backup host. You do not need to connect target volumes to either hosts except for mapping them to a dummy host in SVC environment.



For more information about the SVC environment, refer to “Preparing the storage environment” on page 747. If you use IBM System Storage N series, you do not need specific target volumes, it is managed by the storage device. For more information about snapshot operation on IBM System Storage N series, refer to *IBM System Storage N series*, SG24-7129.

- ▶ **p\_db\_disk**  
Supplementary local disks on the production system. This category contains the DB2 instance directory and the log archive directories (if you use DB2 log management).
- ▶ **b\_disk**  
Local disks for software installation in the backup hosts.
- ▶ **storage pool for TSM**  
Disks or tapes used by TSM to store backup images.

**Note:** If you do not use a full licensed TSM ACS, you do not need a backup system and TSM storage pool. For more information about backup systems, refer to Chapter 2 of the *TSM ACS V5.5 Installation and User's Guide*, SC33-8330 available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

- ▶ **nfs\_disk**  
If you have a separate backup system, or in a DPF environment, all servers need access to one DB2 ACS control files and repository directories, then you need to share the DB2 ACS repository. Normally you would do this with NFS, with disks for NFS mount (**nfs\_disk**). You can either use a separate NFS server or use one production host as NFS server, and others act as NFS clients. Note that, in a DPF environment, you can use one directory under DB2 shared home directory as DB2 ACS repository. We recommend using the following directories for DB2 ACS as shown in Table 14-2.

Table 14-2 DB2 ACS directories

Configuration parameter	Directory	Explanation
ACS_DIR	<DB2 instance home>/acs	Path of DB2 ACS directory
ACS_REPOSITORY	<DB2 instance home>/acs/acsrepository	Path of DB2 ACS repository
VOLUMES_DIR	<DB2 instance home>/acs/acsvolumes	The directory where DB2 ACS target volume files reside.

## 14.2.3 Prerequisites

Before you set up DB2 ACS to do snapshot backups, first make sure that you meet the prerequisites. The following sections list DB2 ACS prerequisites.

### Storage hardware prerequisites

To perform snapshot backup and restore operations, you need a DB2 ACS API driver for your storage device. Table 14-3 lists the integrated DB2 ACS API drivers for the storage hardware for DB2 9.1 Fixpak 1.

Table 14-3 Supported operating systems and hardware in DB2 ACS API

Hardware	AIX with storage area network (SAN) storage	AIX with network file system (NFS) storage	Linux with network file system (NFS) storage
IBM TotalStorage SAN Volume Controller (SVC)	Full support	Not supported	Not supported
IBM Enterprise Storage Server Model 800	Full support except incremental copying is not supported	Not supported	Not supported
IBM System Storage DS6000	Full support except incremental copying is not supported	Not supported	Not supported
IBM System Storage DS8000	Full support except incremental copying is not supported	Not supported	Not supported
IBM System Storage N Series	Fully supported	Fully supported	Fully supported
NetApp V-series	Fully supported	Fully supported	Fully supported

**Note:** DB2 ACS only supports Linux on the following platforms:

- ▶ 64-bit only on x86 (Intel Pentium®, Intel Xeon®, and AMD) processors
- ▶ POWER™ (IBM eServer OpenPower®, System i, or pSeries® systems that support Linux)

## Software prerequisites

Before installing DB2 ACS, make sure that you meet the software prerequisites listed in Table 14-4. The table includes the device drivers needed to attach the supported storage subsystems from which the storage server copy service is to be used to perform the snapshot backup. For more information about host attachment, refer to the related documents of your storage subsystems.

Table 14-4 Software prerequisites

Software	Version	Special considerations
Pegasus Common Interface Model (CIM) server and base providers (available on the AIX Expansion Pack CD)	Use the one in your Expansion Pack CD	Pegasus CIM server and base providers filesets: sysmgt.pegasus.cimserver.rte sysmgt.pegasus.osbaseproviders
Secure Sockets Layer (SSL) package (available on the AIX Toolbox for Linux Applications for POWER Systems CD)	Pegasus CIM server and base providers have requirements for SSL level. Use the same set of CD.	N Series storage server does not require Pegasus CIM server and base providers
libgcc	We recommend the compatible version with your operating system	Only required by N Series storage
NFS	We recommend the compatible version with your operating system	Required when you use NFS. It should be installed by default.
Enterprise Storage Server Copy Services Command-Line Interface (CLI)	2.3 (or later)	Required for IBM Enterprise Storage Server Model 800 disk storage subsystem only
IBM Common Interface Model (CIM) Agent for DS Open (API)	5.1.0.25 (or later)	Required for ESS 800,DS6000, and DS8000 environments only

Software	Version	Special considerations
Host Attachment for SDD on AIX	We recommend the latest version, available at: <a href="http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000106&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en">http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000106&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en</a>	Required for ESS 800,DS6000, DS8000, and SVC environments, as well as IBM System Storage N Series in a SAN environment.
Multipath Subsystem Device Driver (SDD) for AIX	We recommend the latest version, available at: <a href="http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000065&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en">http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000065&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en</a>	Note that you only need SDD or SDDPCM.
Multipath Subsystem Device Driver Path Control Module (SDDPCM) for AIX	We recommend the latest version, available at: <a href="http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000201&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en">http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;uid=ssg1S4000201&amp;loc=en_US&amp;cs=utf-8&amp;lang=en+en</a>	Required for ESS 800,DS6000, DS8000, and SVC environments, as well as IBM System Storage N Series in a SAN environment.
Host Attachment for SDDPCM on AIX	We recommend the latest version, available at: <a href="http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;dc=D400&amp;q1=host+script&amp;uid=ssg1S4000203&amp;loc=en_US&amp;cs=utf-8&amp;lang=en">http://www-1.ibm.com/support/docview.wss?rs=540&amp;context=ST52G7&amp;dc=D400&amp;q1=host+script&amp;uid=ssg1S4000203&amp;loc=en_US&amp;cs=utf-8&amp;lang=en</a>	Note that you only need SDD or SDDPCM.
N Series FCP host attach kit	We recommend the latest version, Consult IBM support to get it	Only required by N Series storage in a SAN environment

**Note:** The above hardware and software prerequisites are for DB2 ACS. For more detailed requirements of TSM ACS V5.5, refer to Chapter 3 of the *TSM ACS V5.5 Installation and User's Guide*, SC33-8330 available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

## File system requirements

In addition to the hardware and software requirements, DB2 ACS has file system requirements too. Refer to File system requirements in Chapter 2 of the *TSM ACS V5.5 Installation and User's Guide*, SC33-8330 available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

## 14.2.4 Installing DB2 Advanced Copy Services

The files and libraries required for DB2 ACS are installed by the IBM DB2 data server installer automatically when the DB2 data server is installed. Before you install the IBM DB2 data server, first you need install the required software.

### Installing prerequisite software

In our test environment, we install the software shown in Table 14-5.

Table 14-5 Installed prerequisite software

Operating system	Software	Version
AIX 5.3 TL07 SP3	libgcc	4.0.0-1
	openssl	0.9.7l-2
	Pegasus CIM server and base providers	sysmgt.pegasus.cimserver.rte 2.5.1.20 sysmgt.pegasus.osbaseproviders 1.2.6.20
SLES 10 SP1	libgcc	4.1.2_20070115-0.11
	openssl	0.9.8a-18.15

After installing the required software from the operating system, make a symbolic link for the **libgcc** on AIX and Red Hat Enterprise Linux (RHEL).

On AIX, run the following command:

```
ln -s /opt/freeware/lib/powerpc-ibm-aix5.3.0/libgcc_s.a  
/usr/lib/libgcc_s.a
```

On RHEL, run the following command:

```
ln -s libssl.so.0.9.7xxx libssl.so.0.9.7  
ln -s libcrypto.so.0.9.7xxx libcrypto.so.0.9.7  
ln -s libssl.so.0.9.7xxx libssl.so  
ln -s libssl.so.0.9.7xxx libssl.so.0
```

## Installing DB2 ACS

You do not need install DB2 ACS separately. When you finish DB2 installation, DB2 ACS is installed automatically.

After installation, we need add a port in `/etc/services` for DB2 ACS agent. In our lab environment, we add the following entries in `/etc/services`:

```
db2acs 57328/tcp # DB2 ACS service port
```

### 14.2.5 Activate DB2 ACS

Before you can use DB2 ACS to perform a snapshot backup for a given database manager instance, you must activate DB2 ACS functionality on that instance.

DB2 provides a `setup.sh` script for activating and configuring DB2 ACS. The script `setup.sh` is located in the `<instance home>/sqlib/acs` directory. Here is the syntax for `setup.sh`:

```
setup.sh -a <action>
        -d <DB2_Instance_Directory>
        -u <Instance_user_ID_name>
        -g <Instance_primary_group_name>
```

Where *action* can be one of these possibilities:

<b>start</b>	Start DB2 ACS daemon
<b>stop</b>	Stop DB2 ACS daemon
<b>query</b>	Query if DB2 ACS is activated in specific DB2 instance
<b>enable</b>	Activate DB2 ACS for specific DB2 instances
<b>disable</b>	Deactivate DB2 ACS for specific DB2 instance

For more information about `setp.sh`, consult the DB2 information center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/c0053160.html>

When a DB2 server instance is created or upgraded, the database manager automatically calls **setup.sh** to activate the DB2 ACS functionality. Example 14-5 shows that we create a DB2 instance **db2snap1** and query the status of DB2 ACS on Linux. It is activated automatically.

*Example 14-5 Create DB2 instance and query DB2 ACS status on Linux*

---

```
mensa:~ # cd /opt/ibm/db2/V9.5/instance
mensa:/opt/ibm/db2/V9.5/instance # ./db2icrt -u db2fenc1 db2snap1
DBI1070I Program db2icrt completed successfully.
mensa:/opt/ibm/db2/V9.5/instance # su - db2snap1
db2snap1@mensa:~> cd sqlib/acs
```

```
db2snap1@mena:~/sql1lib/acs> ./setup.sh -a query -d /home/db2snap1/sql1lib -u
db2snap1 -g db2iadm1
\nchecking /home/db2snap1/sql1lib/acs/acsnnas ...
OK

SUID bit for /home/db2snap1/sql1lib/acs/acsnnas is set. Ok.
```

---

The output on AIX is a little different. Example 14-6 shows the output on AIX.

*Example 14-6 Query DB2 ACS status on AIX*

---

```
[db2snap1@Banda /home/db2snap1/sql1lib/acs]
$ ./setup.sh -a query -d /home/db2snap1/sql1lib -u db2snap1 -g db2iadm1

checking /home/db2snap1/sql1lib/acs/acsnnas ...
OK

checking /home/db2snap1/sql1lib/acs/acsnsan ...
OK

checking /home/db2snap1/sql1lib/acs/acscim ...
OK

SUID bit for /home/db2snap1/sql1lib/acs/acscim is set. Ok.
```

---

You can also activate DB2 ACS manually in case it is disabled occasionally. To activate DB2 ACS manually, run the setup.sh script with appropriate parameters as a user with root authority. Example 14-7 shows the command.

*Example 14-7 Run setup.sh to activate DB2 ACS manually*

---

```
mena:/opt/ibm/db2/V9.5/acs # ./setup.sh -a enable -d /home/db2snap1/sql1lib -u
db2snap1 -g db2iadm1
\nchecking /home/db2snap1/sql1lib/acs/acsnnas ...
OK
```

---

For more information about setup.sh, refer to 14.3, “Recover database command” on page 759

## 14.2.6 Configure DB2 ACS for IBM System Storage N Series with NFS

In this section, we demonstrate how to configure DB2 ACS for IBM System Storage N Series on SLES 10 SP1. In this example, we assume that the following minimum configuration is already in place:

- ▶ Two servers are running SLES 10 SP1 x64.
- ▶ DB2 9.5 FixPack1 with DB2 ACS is activated on the DB2 instance **db2snap1**.
- ▶ An IBM System Storage N5200 storage server is connected to the Linux servers. Volumes shown in Table 14-6 are created and exported on an IBM System Storage N5200 storage server. For information about how to create and export FlexVol® volumes in IBM System Storage N5200, refer to *DB2 9 for UNIX: Integrating with a NetApp Storage System*, available at:

[http://www.netapp.com/us/library/technical-reports/tr\\_3531.html](http://www.netapp.com/us/library/technical-reports/tr_3531.html)

Table 14-6 Volumes and mount points

Purpose	FlexVol Volume	Mount point
DB2 data file system on partition 0 (mensa)	/vol/db2datavol0	/nas/db2data/db2snap1/NODE0000
DB2 log file system on partition 0 (mensa)	/vol/db2logvol0	/nas/db2log/NODE0000
DB2 data file system on partition 1 (lepus)	/vol/db2datavol1	/nas/db2data/db2snap1/NODE0001
DB2 log file system on partition 1 (lepus)	/vol/db2logvol1	/nas/db2log/NODE0001
DB2 home directory in DPF environment (mensa and lepus)	/vol/db2homevol1	/home/db2snap1
DB2 ACS repository	/home/db2snap1/acs/acsrepository Note: This is a directory under DB2 home directory	

Figure 14-3 shows the storage environment in our Lab. For a single partition environment, we use the existing DB2 instance **db2snap1**, which uses the local directory `/home/db2snap1` as home directory. In “DB2 snapshot backups in a partitioned database environment” on page 742, We use the N5200 volume `/vol/db2homevol1` as DB2 home directory `/home/db2snap1`.

**Note:** For IBM System Storage N Series, you do not need to specify target volumes. Snapshots are stored in the same FlexVol volumes.



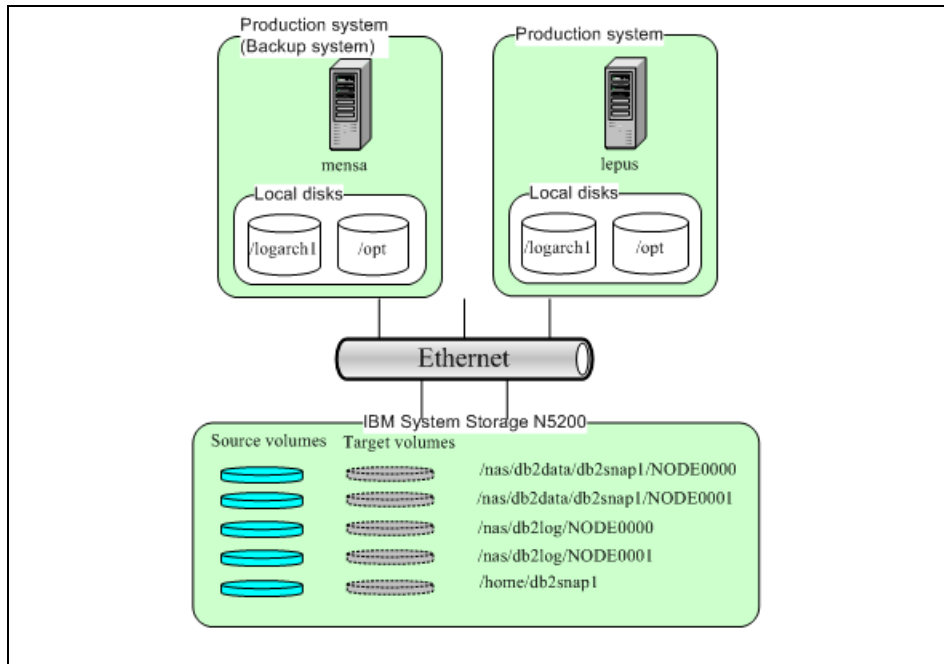


Figure 14-3 Storage environment

## Preparing storage for a single partition database system

We show here how to use NFS to mount storage for DB2 directories on a Linux server with SLES 10 SP1 installed:

1. Edit /etc/hosts file. Add storage server host name and IP address:

```
#NAS storage system
9.43.86.31 itsosj1
```

2. Create a mount point on the Linux server for mensa:

```
mensa:/ # mkdir -p /nas/db2data/db2snap1/NODE0000
mensa:/ # mkdir -p /nas/db2log/NODE0000
```

3. Edit /etc/fstab file to add mount entries:

```
itsosj1:/vol/db2datavo10 /nas/db2data/db2snap1/NODE0000 nfs
hard,rw,nointr,rsize=32768,wsiz=32768,bg,vers=3,tcp 0 0
```

```
itsosj1:/vol/db2logvo10 /nas/db2log/NODE0000 nfs
hard,rw,nointr,rsize=32768,wsiz=32768,bg,vers=3,tcp 0 0
```

4. Mount the file system and change owner to the DB2 instance owner:

```
mensa:/ # mount /nas/db2data/db2snap1/NODE0000
```

```
mena:/ # mount /nas/db2log/NODE0000
mena:/ # chown db2snap1:db2iadm1 /nas/db2data/db2snap1/NODE0000
mena:/ # chown db2snap1:db2iadm1 /nas/db2log/NODE0000
```

## Configuring DB2 ACS

DB2 provides the `setup.sh` script for configuring DB2 ACS. Example 14-8 shows the configuration procedure in our test environment.

### Example 14-8 Configure DB2 ACS through `setup.sh`

---

```
mena:/ # su - db2snap1
db2snap1@mena:~> cd sqllib/acs
db2snap1@mena:~/sqllib/acs> ./setup.sh
\nchecking /home/db2snap1/sqllib/acs/acsnnas ...
OK

Do you have a full TSM license to enable all features of TSM for ACS ?[y/n]

n

***** Profile parameters for section GLOBAL: *****
ACS_DIR [ /home/db2snap1/sqllib/acs ] /home/db2snap1/acs
ACSD [ localhost 57328 ] mensa 57328
TRACE [ NO ]

***** Profile parameters for section ACSD: *****
ACS_REPOSITORY *mandatory parameter* /home/db2snap1/acs/acsrepository

***** Profile parameters for section CLIENT: *****
TSM_BACKUP [ NO ]
MAX_VERSIONS [ ADAPTIVE ]
BK18328E: IBM Tivoli Storage Manager for Advanced Copy Services must be
licensed to set parameter MAX_VERSIONS to a value of ADAPTIVE.

MAX_VERSIONS [ ADAPTIVE ] 2
LVM_FREEZE_THAW [ YES ]
DEVICE_CLASS [ STANDARD ]

***** Profile parameters for section STANDARD: *****

COPYSERVICES_HARDWARE_TYPE *mandatory parameter* NAS_NSERIES
COPYSERVICES_PRIMARY_SERVERNAME *mandatory parameter* itsosj1
COPYSERVICES_USERNAME [superuser ] root

=====

The profile has been successfully created.
Do you want to continue by specifying passwords for the defined devices? [y/n]
```

y

Please specify the passwords for the following profile sections:

STANDARD

master

Creating password file at /home/db2snap1/sqllib/acs/shared/pwd.acsd.

A copy of this file needs to be available to all components that connect to acsd.

BKI1555I: Profile successfully created. Performing additional checks. Make sure to restart all ACS components to reload the profile.

---

Here we describe the following setup.sh parameters:

► ACS\_DIR directory:

The ACS\_DIR directory contains the following subdirectories:

- The subdirectory *logs* contains all logs and trace information that DB2 ACS generates. Check here first when you encounter any problems.
- The subdirectory *shared* is used for information that needs to be shared among all DB2 ACS components. The shared subdirectory currently contains only the password file (pwd.acsd). This file maintains passwords for all devices specified within the profile (see the device section) and a master password, which is used from all components in order to authenticate when connecting to the management agent.

The default DB2 ACS directory is <instance home>/sqllib/acs. We suggest that you create a different directory under DB2 instance home directory, such as <instance home>/acs, because if you update or drop the DB2 instance, the logs and shared files under <instance home>/sqllib/acs are then removed. In a DPF environment, <instance home> is a NFS directory. All database partitions can share this directory.

- ACSD is the master server host name and port. In order to use DB2 ACS with multiple instances of DB2 on a server, you can specify different ports in **setup.sh** during setup. In this example, we use mensa 57328. We recommend that you use a host name other than localhost.

**Note:** In “Installing DB2 ACS” on page 732, we add 57328 to /etc/services as the DB2 ACS agent port. In case you have multiple instances of DB2 on a server, we suggest that you add all ports used by DB2 ACS in your system to /etc/services.

- ▶ ACS\_REPOSITORY is the directory on the backup server that keeps a record of all backups taken. The ACS repository should be placed on a shared path for all machines running the DB2 instance. This path, however, cannot be a part of any of the volumes that are to be part of the backup; whether it is a log or data volume. We suggest using <ACS\_DIR>/acsrepository for DB2 ACS.
- ▶ MAX\_VERSIONS is the maximum number of backups that can be taken using the Integrated snapshot backup. For DB2 ACS, it can be set to a maximum of 2. If a full license TSM ACS is installed, then an unlimited number of backups can be taken if MAX\_VERSIONS is set to ADAPTIVE.

For the remaining setup parameters, it is dependent on the type of hardware used to perform FlashCopy or snapshot:

- ▶ COPYSERVICES\_HARDWARE\_TYPE is the hardware type of the disks used. For IBM System Storage N Series with NFS, this is NAS\_NSERIES. Other valid values are DS8000, DS6000, ESS800, SVC, and SAN\_NSERIES. Only one system can be specified.
- ▶ COPYSERVICES\_PRIMARY\_SERVERNAME defines the TCP/IP address of the host running CIM agent for DS open API (DS6000, DS8000 and ESS800), the SVC master console, or the filer for IBM System Storage N Series filer. In this example, it is itsosj1. Note that you can specify either server name or address. When you use a server name, make sure it is pingable from the backup host.
- ▶ COPYSERVICES\_USERNAME is a user name on the copy services servers that can make FlashCopy or snapshot. We use root on the NAS filer in this example.

Following the STANDARD section is the password section.

STANDARD is the password used for a copy services user specified by COPYSERVICES\_USERNAME. In our example, we input the password for root on the NAS filer. MASTER, on the other hand, is the password used in communication within the ACS services, and can be set to anything.

The configuration file profile is under the <instance home>/sqllib/acs directory, which is configured with **setup.sh**. Example 14-9 shows the content in this example. You can modify this file directly to configure DB2 ACS too.

*Example 14-9 DB2 ACS profile*

---

```
db2snap1@mensa:~/sqllib/acs> cat profile
>>> GLOBAL
ACS_DIR /home/db2snap1/acs
ACSD mensa 57328
# TRACE NO
<<<
>>> ACSD
```

```

ACS_REPOSITORY /home/db2snap1/acs/acsrepository
<<<
>>> CLIENT
# TSM_BACKUP NO
MAX_VERSIONS 2
# LVM_FREEZE_THAW YES
# DEVICE_CLASS STANDARD
<<<
>>> STANDARD
COPYSERVICES_HARDWARE_TYPE NAS_NSERIES
COPYSERVICES_PRIMARY_SERVERNAME itsosj1
COPYSERVICES_USERNAME root
<<<

```

---

When the DB2 ACS is configured, check if the agent is started as shown in Example 14-10.

*Example 14-10 Verify if DB2 ACS agent is started*

```

db2snap1@mena:~/sqllib/acs> ps -ef |grep acs |grep -v grep
db2snap1 16274      1  0 15:27 ?                00:00:00 /home/db2snap1/sqllib/acs/acsd
root      17082      1  0 15:32 ?                00:00:00 /home/db2snap1/sqllib/acs/acsnas -D
db2snap1@mena:~/sqllib/acs> netstat -an |grep 57328
tcp        0      0 0.0.0.0:57328          0.0.0.0:*              LISTEN
tcp        0      0 127.0.0.1:57328        127.0.0.1:41453        ESTABLISHED
tcp        0      0 127.0.0.1:41453        127.0.0.1:57328        ESTABLISHED
tcp        0      0 :::57328               :::*                    LISTEN

```

---

## Offline backup and restore with snapshot

To demonstrate the offline database backup and restore with snapshot, we create a database TEST on /nas/db2data and change the database log path to /nas/db2log as shown in Example 14-11.

*Example 14-11 Create database and change log directory*

```

db2snap1@mena:~> db2 "create db test on /nas/db2data"
DB20000I The CREATE DATABASE command completed successfully.
db2snap1@mena:~> db2 "update db cfg for test using newlogpath /nas/db2log"
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2snap1@mena:~> db2 connect to test

```

### Database Connection Information

```

Database server      = DB2/LINUX8664 9.5.1
SQL authorization ID = DB2SNAP1
Local database alias = TEST

```

```

db2snap1@mena:~> db2 connect reset
DB20000I The SQL command completed successfully.

```

---

**Note:** Before performing the snapshot backup, make sure that both the database data and log are on a storage system that supports the snapshot backup. Otherwise, it fails with an error. In our example, if we use a local directory /nas/db2log1 as log path, then we encounter the following error:

```
SQL2062N An error occurred while accessing media "libacbdb2.so".
Reason code: "18".
```

You can check DB2 ACS log for further diagnostics, which is located under <ACS\_DIR>/logs. The file is acsd.<date>.log. In our example, the error is: “DEV 0001 04/23/08 11:35:02 IDS6927E: Failed to find N-Series volume for file '/nas/db2log1/NODE0000/'. Error: EEP3411E Invalid arguments were passed to function dmMapNfsVolume.”

### **Backup database**

Example 14-12 shows how to take an offline DB2 backup with snapshot. We use the **db2acsutil** command to obtain the details about the backup.

#### *Example 14-12 Backup database*

---

```
db2snap1@mensa:~> db2 backup db test use snapshot
```

Backup successful. The timestamp for this backup image is : 20080416161641

```
db2snap1@mensa:~> db2acsutil query snapshot db test show details
```

```
Instance : db2snap1
Database : TEST
Partition : 0
Image timestamp : 20080416161641
Host : mensa
Owner :
DB2 Version : 9.5.1
Creation time : Wed Apr 16 16:16:43 2008
First active log (chain:file) : 0:0
Metadata bytes : 6196
Progress state : Successful
Usability state : Remotely mountable + Repetitively restorable +
Destructively restorable
Bytes completed : 0
Bytes total : 0
```

---

**Note:** **db2acsutil** is a utility to manage DB2 snapshot backup objects. For detailed usage information, refer to DB2 information center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.admin.cmd.doc/doc/r0052775.html>

In Example 14-13 we connect to TEST to create one table test and insert two rows. These transactions are for verifying the database restore later.

*Example 14-13 Create table and insert two rows*

---

```
db2snap1@mena:~> db2 connect to test

      Database Connection Information

Database server          = DB2/LINUX8664 9.5.1
SQL authorization ID    = DB2SNAP1
Local database alias    = TEST

db2snap1@mena:~> db2 "create table test (id int, name char(10))"
DB20000I The SQL command completed successfully.
db2snap1@mena:~> db2 "insert into test values (1, 'snap1')"
DB20000I The SQL command completed successfully.
db2snap1@mena:~> db2 "insert into test values (2, 'snap2')"
DB20000I The SQL command completed successfully.
db2snap1@mena:~> db2 connect reset
DB20000I The SQL command completed successfully.
```

---

**Restore database**

We restore the database TEST with the **restore db <dbname> use snapshot** command, then verify the restore by checking if the table created after the backup exists in the database. Example 14-14 shows that database TEST is restored to the point where we do snapshot backup.

*Example 14-14 Restore database TEST from snapshot backup*

---

```
db2snap1@mena:~> db2 restore db test use snapshot
SQL2539W Warning! Restoring to an existing database that is the same as the
backup image database. The database files will be deleted.
Do you want to continue ? (y/n) y
DB20000I The RESTORE DATABASE command completed successfully.
db2snap1@mena:~> db2 connect to test
```

```
      Database Connection Information

Database server          = DB2/LINUX8664 9.5.1
SQL authorization ID    = DB2SNAP1
Local database alias    = TEST

db2snap1@mena:~> db2 list tables for schema db2snap1

Table/View      Schema      Type      Creation time
-----
0 record(s) selected.
```

---

## Online backup with snapshot

To take an online backup, the database must be enabled in archive log mode by updating the database configuration parameter. Example 14-15 shows how we enable online backup. Note that DB2 requires you take a database backup after the archiving logging is enabled.

### *Example 14-15 Enable online backup*

---

```
mensa:/ # mkdir /logarch1
mensa:/ # chown db2inst1:db2iadml /logarch1
mensa:/ # su - db2snap1
db2snap1@mensa:~> db2 update db cfg for test using logarchmeth1 disk:/logarch1
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
```

---

Take the online snapshot backup by using the following command:

```
db2snap1@mensa:~> db2 backup db test online use snapshot
Backup successful. The timestamp for this backup image is :
20080422111526
```

By default, the online backup includes logs too. To exclude logs in the backup, use the following command:

```
db2snap1@mensa:~> db2 backup db test online use snapshot exclude logs
Backup successful. The timestamp for this backup image is :
20080422111920
```

## DB2 snapshot backups in a partitioned database environment

DB2 snapshot backup is also supported in a partitioned database environment. The system preparation tasks for a partitioned database environment are about the same as for a single partitioned database environment:

- ▶ Prepare all the servers to meet prerequisites as described in 14.2.3, “Prerequisites” on page 728.
- ▶ Then, install and activate DB2 ACS on all servers by following the procedures described in 14.2.4, “Installing DB2 Advanced Copy Services” on page 731 and 14.2.5, “Activate DB2 ACS” on page 732.
- ▶ Mount the NFS file system. “Preparing storage for a single partition database system” on page 735 shows how to mount NFS file systems on SLES 10 SP1.



Table 14-7 lists file systems for our example.

Table 14-7 DPF lab environment in our example

Server	mensa	lepus
Shared home file system	/home/db2snap1	/home/db2snap1
Data file system	/nas/db2data/db2snap1/ NODE0000	/nas/db2data/db2snap1/ NODE0001
Log file system	/nas/db2log/NODE0000	/nas/db2log/NODE0001
Database partition	0	1

Example 14-16 is a simple test showing that our example DB2 instance for a partitioned database is working.

Example 14-16 Make sure DB2 instance works properly

---

```
db2snap1@mensa:~/sql1lib> cat db2nodes.cfg
0 mensa 0 mensa_int
1 lepus 0 lepus_int
db2snap1@mensa:~/sql1lib> db2start
04/21/2008 13:59:48    0  0  SQL1063N  DB2START processing was successful.
04/21/2008 13:59:49    1  0  SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

---

### Configuring DB2 ACS

On a partitioned database environment, the DB2 ACS configuration script `setup.sh` should be run on the DB2 catalog partition. The steps shown in “Configuring DB2 ACS” on page 736 can be applied for configuring DB2 ACS for a partitioned database environment.

### Back up database

To demonstrate the database backup and restore, we create a database TEST with one table, test, and insert some data into the table. See Example 14-17.

Example 14-17 Create database, table, and insert data

---

```
db2snap1@mensa:~/sql1lib> db2 "create table test (id int, name char(10))"
DB20000I  The SQL command completed successfully.
db2snap1@mensa:~/sql1lib> db2 update db cfg for test using newlogpath
/nas/db2log
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
db2snap1@mensa:~/sql1lib> db2 "insert into test values(1,'snap1')"
DB20000I  The SQL command completed successfully.
db2snap1@mensa:~/sql1lib> db2 "insert into test values(2,'snap2')"
DB20000I  The SQL command completed successfully.
db2snap1@mensa:~/sql1lib> db2 "insert into test values(3,'snap3')"
```

```
.....
ddb2snap1@mena:~/sqllib> db2 "select dbpartitionnum(id) as PartNm, id from
test order by dbpartitionnum(id)"
```

PARTNM	ID
0	5
0	10
0	11
1	1
1	2
1	3
1	4
1	12

8 record(s) selected.

---

In a partitioned database environment, you can back up the database one partition at a time, or use *single system view (SSV) backup* to back up all partitions with snapshot. Example 14-18 shows how we back up database catalog partition 0 on mensa.

*Example 14-18 Backup catalog partition on mensa*

---

```
db2snap1@mena:~> db2 backup db test use snapshot
```

Backup successful. The timestamp for this backup image is : 20080421151043

SQL2431W The database backup succeeded. On each database partition, only those log files that were active during the backup operation are included in the backup image.

```
db2snap1@mena:~> db2acsutil query snapshot db test show details
```

```
Instance : db2snap1
Database : TEST
Partition : 0
Image timestamp : 20080421151043
Host : mensa
Owner :
DB2 Version : 9.5.1
Creation time : Mon Apr 21 15:10:45 2008
First active log (chain:file) : 0:0
Metadata bytes : 6196
Progress state : Successful
Usability state : Remotely mountable + Repetitively restorable +
Destructively restorable
Bytes completed : 0
Bytes total : 0
```

---

Example 14-19 shows how we back up all database partitions on mensa.

*Example 14-19 Backup all database partitions on mensa*

---

```
db2snap1@mensa:~> db2 "backup db test on all dbpartitionnums use snapshot"
```

```
Part Result
```

```
-----  
0000 DB20000I The BACKUP DATABASE command completed successfully.  
0001 DB20000I The BACKUP DATABASE command completed successfully.
```

Backup successful. The timestamp for this backup image is : 20080421151921

```
db2snap1@mensa:~> db2acsutil query snapshot db test taken at 20080421151921  
show details
```

```
Instance : db2snap1  
Database : TEST  
Partition : 1  
Image timestamp : 20080421151921  
Host : lepus  
Owner :  
DB2 Version : 9.5.1  
Creation time : Mon Apr 21 15:19:55 2008  
First active log (chain:file) : 0:0  
Metadata bytes : 6196  
Progress state : Successful  
Usability state : Remotely mountable + Repetitively restorable +  
Destructively restorable  
Bytes completed : 0  
Bytes total : 0
```

```
Instance : db2snap1  
Database : TEST  
Partition : 0  
Image timestamp : 20080421151921  
Host : mensa  
Owner :  
DB2 Version : 9.5.1  
Creation time : Mon Apr 21 15:19:24 2008  
First active log (chain:file) : 0:0  
Metadata bytes : 6196  
Progress state : Successful  
Usability state : Remotely mountable + Repetitively restorable +  
Destructiv
```

---

### ***Restore database***

When restoring a database with an SSV backup in a multi-partitioned database environment, we can use **db2\_a11** to restore the database on the partitions parallel. Example 14-20 shows how we restore database TEST with the SSV backup image taken in Example 14-19.

*Example 14-20 Restore SSV backup image parallel in a multi-partitioned database environment*

---

```
db2snap1@mensa:~> db2_a11 " | db2 restore db test use snapshot taken at
20080421151921 without prompting"
```

```
rah: primary monitoring process for db2 is 9196
```

```
mensa: SQL2540W Restore is successful, however a warning "2539" was encountered
mensa: during Database Restore while processing in No Interrupt mode.
mensa: db2 restore db test ... completed rc=2
```

```
lepus: SQL2540W Restore is successful, however a warning "2539" was encountered
lepus: during Database Restore while processing in No Interrupt mode.
lepus: db2 restore db test ... completed rc=2
```

---

The SQL2539W warning received during database restore operation is due to our attempting to overwrite the existing database TEST:

SQL2539W Warning! Restoring to an existing database that is the same as the backup image database. The database files will be deleted.

DB2 logs the backup and restore information in a history file. Example 14-21 illustrates that the history file shows that restore operation is successful on both servers.

*Example 14-21 Check restore operation result*

---

```
db2snap1@mensa:~> db2 "list history backup since 20080421152021 for test"
```

```
List History File for test
```

```
Number of matching file entries = 1
```

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
R D 20080421154138001 F F S0000000.LOG S0000000.LOG 20080421151921
-----
```

```
Contains 2 tablespace(s):
```

```
00001 SYSCATSPACE
00002 USERSPACE1
-----
```

```
Comment: RESTORE TEST NO RF
Start Time: 20080421154138
End Time: 20080421154141
Status: A
-----
```

```
EID: 5 Location:
db2snap1@lepus:~> db2 "list history backup since 20080421152021 for test"
```

### List History File for test

Number of matching file entries = 1

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
R D 20080421154209001 F F S0000000.LOG S0000000.LOG 20080421151921
-----
Contains 1 tablespace(s):

00001 USERSPACE1
-----
Comment: RESTORE TEST NO RF
Start Time: 20080421154209
End Time: 20080421154213
Status: A
-----
EID: 3 Location:
```

---

## 14.2.7 Configure DB2 ACS for IBM System Storage SVC

In this section, we demonstrate how to configure DB2 ACS for IBM SVC in a partitioned database environment.

### Preparing the storage environment

When you prepare your storage environment for IBM SVC, consider the following characteristics, requirements, and limitations:

- ▶ SVC Master Console (MC) version 4.2.0.616 is required.
- ▶ Volumes in the SVC are referred to as virtual disks (vDisks) and are addressed by a LUN. The user can define the virtual disks in the Web GUI of the SVC master console.
- ▶ In the SVC, FlashCopy occurs between a source virtual disk and a target virtual disk.
- ▶ The virtual disks must have the same size.
- ▶ A virtual disk is designated by a name, which must be unique.
- ▶ The minimum granularity that SVC supports for FlashCopy is an entire virtual disk. The FlashCopy of part of a virtual disk is not supported.
- ▶ The source and target virtual disks must both be managed by the same SVC cluster, but they can be in different I/O groups within that cluster.
- ▶ The source and target virtual disks are available (almost) immediately after the FlashCopy is initiated.

- ▶ After the start of the FlashCopy, the target and source can be updated independently.
- ▶ SVC FlashCopy associates a source virtual disk and a target virtual disk together in a FlashCopy mapping.
- ▶ Each virtual disk can be a member of only one FlashCopy mapping, and a FlashCopy mapping always has exactly one source and one target virtual disk. Therefore, it is not possible for a virtual disk to simultaneously be the source for one FlashCopy mapping and the target for another.
- ▶ The sizes of vDisks that are members of a FlashCopy mapping cannot be changed while the mapping is in effect.
- ▶ An SVC supports the creation of enough FlashCopy mappings to allow each virtual disk to be a member of a FlashCopy mapping.
- ▶ Incremental FlashCopy is supported by the SVC starting with version 4.2.1. This SVC version is not supported by DB2 9.5 FixPak 1.
- ▶ The use of multiple target volumes associated with the same source volume is supported by the SVC starting from version 4.2.1. This SVC version is not supported by DB2 9.5 FixPak 1.
- ▶ The SVC requires that SDD or SDDPCM be installed on all attached hosts.

Note that you are not required to define FlashCopy mappings for source and target volumes in SVC console. DB2 ACS does it automatically when you do snapshot backup, and DB2 deletes it automatically after FlashCopy finishes. Therefore, you can have multiple snapshot backups even if DB2 ACS does not yet support multiple target volumes associated with the same source volume.

In this scenario, we assume that SVC is already there and eight vDisks are created. For more information about SVC configuration, refer to *IBM System Storage SAN Volume Controller*, SG24-6423.

Figure 14-4 shows the storage environment in our lab. Source volumes used for the database must map to the corresponding production hosts, In this example, vDisks FC\_HADR0001 and FC\_HADR0002 are mapped to Banda for database partition 0. vDisks FC\_HADR\_05 and FC\_HADR\_06 are mapped to Baltic for database partition 1. Target volumes must map to a backup host or a dummy host if no backup host exists, In this example, we create a dummy host DB2ACS in the SVC console and map FC\_HADR0003, FC\_HADR0004, FC\_HADR\_07, and FC\_HADR\_08 to the dummy host. Note that you must specify a non-existing WWPN in your SAN environment when you create the dummy host.

Target volumes must map to a backup host or a dummy host if no backup host exists, In this example, we create a dummy host DB2ACS in the SVC console and map FC\_HADR0003, FC\_HADR0004, FC\_HADR\_07, and FC\_HADR\_08 to the dummy host. Note that you must specify a non-existing WWPN in your SAN environment when you create the dummy host.

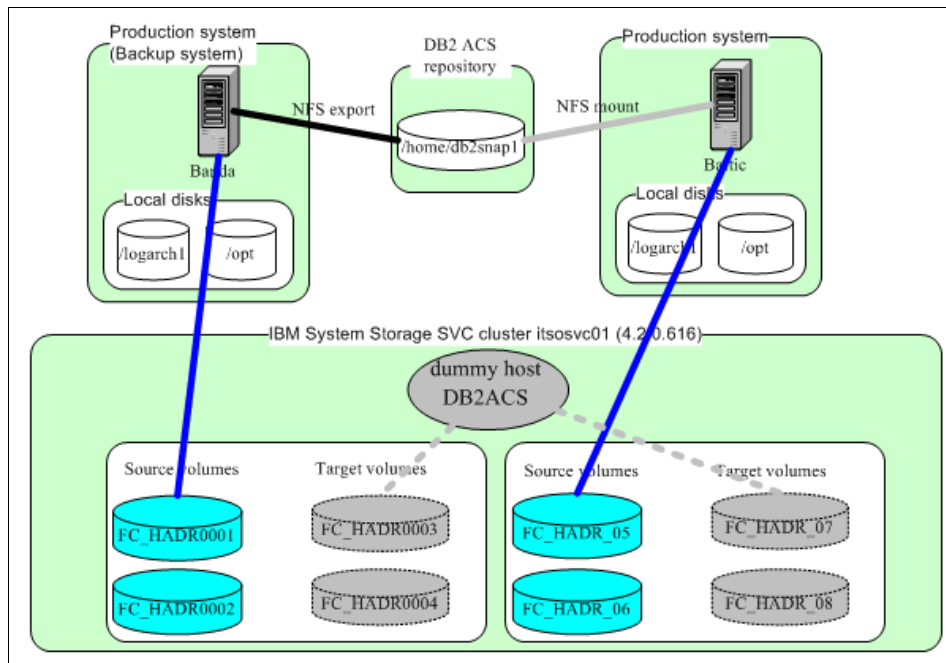


Figure 14-4 SVC storage environment

Figure 14-5 shows that we create a dummy host DB2ACS in the Web GUI of SVC master console.

The screenshot displays the configuration interface for a dummy host. The fields are as follows:

- Host Name:** DB2 ACS
- Type:** Generic
- Port Mask (1 allows access, 0 denies access):** 1111
- \*I/O Groups:** A list containing io\_grp0, io\_grp1, io\_grp2, and io\_grp3, with io\_grp0 selected.
- Available Ports:** A list containing WWPNs 210000E08B8934CE, 210000E08B18208B, and 10000000C932A8AA, with 210000E08B8934CE selected.
- Selected Ports:** An empty list.
- Buttons:** Add > and < Remove.
- Additional Ports (type in WWPNs, one per line):** A text area containing 10000000C9FFFFFF.
- Buttons:** OK and Cancel.

Figure 14-5 Create dummy host DB2ACS



Figure 14-6 shows vDisk mappings in the Web GUI of SVC master console.

Select	Host	vDisk	SCSI ID
<input type="radio"/>	BALTIC	FC_HADR_05	0
<input type="radio"/>	BALTIC	FC_HADR_06	1
<input type="radio"/>	Banda	FC_HADR0001	0
<input type="radio"/>	Banda	FC_HADR0002	1
<input type="radio"/>	DB2ACS	FC_HADR0004	1
<input type="radio"/>	DB2ACS	FC_HADR0003	0
<input type="radio"/>	DB2ACS	FC_HADR_07	2
<input type="radio"/>	DB2ACS	FC_HADR_08	3

Figure 14-6 vDisks mappings

Table 14-8 shows volume groups (VGs), logical volumes (LVs), and mount points we use in this example. Note that DB2 ACS only supports JFS2. All database file systems must be JFS2. You need to enable `cio` when mounting the file systems too.

Table 14-8 Volume groups, logical volumes, and mount points

Purpose	vDisk	VG	LV	Mount point
Data (Banda)	FC_HADR0001	svcdatavg0	fslv00	/svc/db2data/db2snap1/ NODE0000
Log (Banda)	FC_HADR0002	svcllogvg0	fslv01	/svc/db2log/NODE0000
Data (Baltic)	FC_HADR_05	svcdatavg1	fslv00	/svc/db2data/db2snap1/ NODE0001
Log (Baltic)	FC_HADR_06	svcllogvg1	fslv01	/svc/db2log/NODE0001
Target volumes (DB2ACS)	FC_HADR0003			
	FC_HADR0004			
	FC_HADR_07			
	FC_HADR_08			

Example 14-22 shows the entries in /etc/filesystems in server Banda.

*Example 14-22 /etc/filesystems entries*

---

```
/svc/db2data/db2snap1/NODE0000:
    dev           = /dev/fs1v00
    vfs           = jfs2
    log           = /dev/log1v00
    mount         = true
    options       = cio,rw
    account       = false
/svc/db2log/NODE0000:
    dev           = /dev/fs1v01
    vfs           = jfs2
    log           = /dev/log1v01
    mount         = true
    options       = cio,rw
    account       = false
```

---

## Configuring CIM Object Manager and SVC Master Console

CIM Agent is the component used to communicate between DB2 ACS and the storage subsystems ESS 800, DS6000, DS8000, and SVC. Figure 14-1 on page 725 shows the operating environment using CIMOM.

For SVC, the CIM Agent is shipped as part of the SVC storage subsystem, and is installed on SVC Master Console (MC) already. DB2 ACS supports non-SSL connectivity, and SSL connectivity with NO\_CERTIFICATE option to CIM agent. We suggest that you use SSL connectivity to the CIM agent.

If you want to use HTTPS connectivity, you must do the following steps to enable null trust provider mode of CIMOM service:

1. Log on to SVC MC server, use **Start** → **Administrative Tools** → **Services** to start Services control panel, and stop the following service:

IBM CIM Object Manager - SVC

2. Edit the cimom.properties file located at the CIM Agent installation directory (default installation directory is C:\Program Files\IBM\svconconsole\cimom) and change the properties as follows:

DigestAuthentication=False

3. Restart the following service:

IBM CIM Object Manager - SVC

If you want to use HTTP connectivity, you must enable HTTP service (default HTTP port 5988) first. Follow these steps to enable the HTTP connectivity:

1. Log on to the SVC MC server, use **Start** → **Administrative Tools** → **Services** to start Services control panel, and stop the following services:

- IBM CIM Object Manager - SVC
- IBM WebSphere Application Server V6 - SVC

2. Edit cimom.properties file located at the CIM Agent installation directory (default installation directory is C:\Program Files\IBM\svconconsole\cimom) and change the properties as follows:

```
ServerCommunication=HTTP
Port=5988
DigestAuthentication=False
```

3. Change ICAConsole and SVCConsole application settings to use non-SSL connectivity:

Add or change, if already there, the following lines in the cimon.properties file:

```
Port=5988
SslEnabled=false
```

Change the cimon.properties file under the following directory:

```
C:\Program Files\IBM\svconconsole\console\embeddedWAS\profiles\SVCCProfile\
installedApps\SVCCell\ICAConsole.ear\ICAConsole.war\WEB-INF
```

```
C:\Program Files\IBM\svconconsole\console\embeddedWAS\profiles\SVCCProfile\
installedApps\SVCCell\SVCConsole.ear\SVCConsole.war\WEB-INF
```

```
C:\Program Files\IBM\svconconsole\console\embeddedWAS\profiles\SVCCProfile\
config\cells\SVCCell\applications\ICAConsole.ear\deployments\ICAConsole\
ICAConsole.war\WEB-INF
```

```
C:\Program Files\IBM\svconconsole\console\embeddedWAS\profiles\SVCCProfile\
config\cells\SVCCell\applications\SVCConsole.ear\deployments\SVCConsole\
SVCConsole.war\WEB-INF
```

4. Restart the following services:

- IBM CIM Object Manager - SVC
- IBM WebSphere Application Server V6 - SVC

## Configuring DB2 ACS

In this section, we describe how to configure DB2 ACS with **setup.sh**. Make sure that the DB2 partitioned database instance is ready to use and start it.

Run **setup.sh** to configure DB2 ACS to use SVC devices for taking a snapshot backup. Example 14-23 shows the configuration procedure in this example.

*Example 14-23 Configure DB2 ACS to use SVC devices*

---

```
$ ./setup.sh

checking /home/db2snap1/sql/lib/acs/acsnas ...
OK

checking /home/db2snap1/sql/lib/acs/acnsan ...
OK

checking /home/db2snap1/sql/lib/acs/acscim ...
OK

Do you have a full TSM license to enable all features of TSM for ACS ?[y/n]

n

***** Profile parameters for section GLOBAL: *****
ACS_DIR  [/home/db2snap1/sql/lib/acs ] /home/db2snap1/acs
ACSD     [localhost 57328 ]      Banda 57328
TRACE   [NO ]

***** Profile parameters for section ACSD: *****
ACS_REPOSITORY *mandatory parameter* /home/db2snap1/acs/acsrepository

***** Profile parameters for section CLIENT: *****
TSM_BACKUP      [NO ]
MAX_VERSIONS    [ADAPTIVE ]    2
LVM_FREEZE_THAW [YES ]
DEVICE_CLASS    [STANDARD ]

***** Profile parameters for section STANDARD: *****

COPYSERVICES_HARDWARE_TYPE *mandatory parameter* SVC
COPYSERVICES_PRIMARY_SERVERNAME [localhost ] itsosvc01
VOLUMES_DIR *mandatory parameter* /home/db2snap1/acs/acsvolumes
COPYSERVICES_USERNAME [superuser ]
COPYSERVICES_COMMPROTOCOL [HTTPS ]
COPYSERVICES_CERTIFICATEFILE [NO_CERTIFICATE]
COPYSERVICES_SERVERPORT [5999]
COPYSERVICES_TIMEOUT [6 ]
SVC_COPY_RATE [80 ]
=====

The profile has been successfully created.
Do you want to continue by specifying passwords for the defined devices? [y/n]
```

y

Please specify the passwords for the following profile sections:

STANDARD

master

Creating password file at /home/db2snap1/sql/lib/acs/shared/pwd.acsd.

A copy of this file needs to be available to all components that connect to acsd.

BKI1555I: Profile successfully created. Performing additional checks. Make sure to restart all ACS components to reload the profile.

---

Most of the input parameters are described in “Configuring DB2 ACS” on page 736. Following are specific parameters required for SVC devices:

- ▶ **COPYSERVICES\_HARDWARE\_TYPE** is the hardware type of the disks used. For IBM System Storage SVC, this is SVC.
- ▶ **COPYSERVICES\_PRIMARY\_SERVERNAME** is the host name or IP address of the IBM System Storage SVC MC, which runs CIMOM object manager service. In this example, it is itsosvc01. Make sure it is defined in the /etc/hosts files.
- ▶ **VOLUMES\_DIR** is the directory containing target volumes files. It should be in an NFS mount directory and accessible for all DB2 partitions. In this example, we use /home/db2snap1/acs/acsvolumes. You have to specify target volume files for each partition. The file name rule is <DB2 instance>.<database name>.<class name>.<database partition>.fct. In our example, we defined two files for two database partitions. Example 14-24 and Example 14-25 show the file names and contents.

*Example 14-24 db2snap1.TEST.STANDARD.NODE0000.fct*

---

```
>>> volumes_set_1
TARGET_VOLUME FC_HADR0003 - -
TARGET_VOLUME FC_HADR0004 - -
<<< volumes_set_1
```

---

*Example 14-25 db2snap1.TEST.STANDARD.NODE0001.fct*

---

```
>>> volumes_set_1
TARGET_VOLUME FC_HADR_07 - -
TARGET_VOLUME FC_HADR_08 - -
<<< volumes_set_1
```

---

In this example, we only define one volumes set. The statements, >>> volumes\_set\_x and <<< volumes\_set\_x, define the volume set name, where x is a number to distinguish different volume sets. Between these two directives, TARGET\_VOLUME specifies source/target volumes pair. There are three columns followed TARGET\_VOLUME, the first column is the target volume serial number. For SVC, it is vDisk name. The second column is the source volume serial number. The third column is the volume size. Note that the target volume must be in the same storage subsystem as the source volume. Generally you do not have to define the source volume serial number and volume size, just specify with "-". DB2 ACS detects the source volumes and fills them automatically.

- ▶ COPYSERVICES\_USERNAME is a user name on the IBM System Storage SVC for FlashCopy services. We use *superuser* in this example.
- ▶ COPYSERVICES\_COMMPROTOCOL is the protocol used by DB2 ACS to communicate with CIMOM object manager running on the storage subsystems. It has two values, HTTPS or HTTP. We recommend using HTTPS. IBM System Storage SVC currently supports HTTP and HTTPS with NO\_CERTIFICATE option.
- ▶ COPYSERVICES\_SERVERPORT is the service port for the CIMOM object manager on the storage subsystems. The default port number depends on the settings of COPYSERVICES\_HARDWARE\_TYPE and COPYSERVICES\_COMMPROTOCOL. The default HTTPS port for DS8000, DS6000, and ESS800 is 5989. The default HTTPS port for SVC is 5999. For HTTP protocol, for all types of devices, the default port is 5988.
- ▶ COPYSERVICES\_TIMEOUT is the maximum amount of time (in minutes) the CIM client waits for the response to a call issued to the CIMOM (CIM Agent). If the CIM client does not receive a response within this time, an error message is issued. The default value is 6.
- ▶ SVC\_COPY\_RATE specifies the priority that the SVC gives to the FlashCopy background process for the current backup or restore.

For more detailed description of these parameters, refer to Chapter 6 of the *TSM ACS V5.5 Installation and User's Guide*, SC33-8330 available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

Before you do a snapshot backup and restore, make sure **acsd** and **acscim -D** are started successfully; you can use **ps -ef |grep acs** to check them. Note that **acscim -D** exits for 5 minutes a time, then **init** respawns it. You might not see it from the output. But make sure your `etc/inittab` has the following entries:

```
ac00:2345:respawn:/home/db2snap1/sql1lib/acs/acscim -D
ac01:2345:respawn:/home/db2snap1/sql1lib/acs/acsd
```

If these processes are not started automatically, try using the following command to start them:

```
<instance home>/sqllib/acs/setup.sh -a start -u <instance owner> -g <instance owner group>
```

These two processes are critical for taking a snapshot backup. You must make sure that they work before you perform the snapshot backup.

## Performing snapshot backup and restore

Now we create a database and perform snapshot backup and restore. Example 14-26 shows the script we use to create the database TEST. We use DMS for catalog table space and user table space, which resides on a JFS2 file system with **cio** enabled on SVC vDisks. We also update NEWLOGPATH to a JFS2 file system with **cio** enabled on SVC vDisks and enable archive log mode. Note that you can also use SMS table space as described in “DB2 snapshot backups in a partitioned database environment” on page 742.

### *Example 14-26 Scripts to create database TEST*

---

```
create database test on /svc/db2data
catalog tablespace managed by database using
(file '/svc/db2data/db2snap1/NODE000 $N /container/sys.dat' 250000)
user tablespace managed by database using
(file '/svc/db2data/db2snap1/NODE000 $N /container/user.dat' 250000);
update db cfg for test using newlogpath /svc/db2log;
update db cfg for test using logretain on;
backup db test on all dbpartitionnums to /dev/null;
activate db test;
deactivate db test;
```

---

Example 14-27 shows how a snapshot backup on all database partitions is taken successfully.

### *Example 14-27 Taking snapshot backup*

---

```
$ db2 backup db test on all dbpartitionnums use snapshot
Part Result
-----
0000 DB20000I The BACKUP DATABASE command completed successfully.
0001 DB20000I The BACKUP DATABASE command completed successfully.
```

```
Backup successful. The timestamp for this backup image is : 20080501145123
```

---

You can check FlashCopy status from SVC console. Figure 14-7 shows that four FlashCopy mappings are copying, which belong to two consistency groups. One consistency group is for database partition0; the other one is for database partition1.

Select	Name	State	Source VDisk	Target VDisk	FC Group	Progress (%)	Priority	Automatically
<input type="radio"/>	fmap0	Copying	FC_HADR_06	FC_HADR_07	TFFPYIXDA79395		9	80 Disabled
<input type="radio"/>	fmap1	Copying	FC_HADR_05	FC_HADR_08	TFFPYIXDA79395		9	80 Disabled
<input type="radio"/>	fmap2	Copying	FC_HADR0001	FC_HADR0003	TFFPYIXIG5139B		8	80 Disabled
<input type="radio"/>	fmap3	Copying	FC_HADR0002	FC_HADR0004	TFFPYIXIG5139B		8	80 Disabled

Page 1 of 1      Total: 4   Filtered: 4   Displayed: 4   Selected: 0

Figure 14-7 FlashCopy mappings

Figure 14-8 shows that two consistency groups are copying.

Select	Name	State
<input type="radio"/>	TFFPYIXDA79395	Copying
<input type="radio"/>	TFFPYIXIG5139B	Copying

Page 1 of 1      Total: 2   Filtered: 2   Displayed: 2   Selected: 0

Figure 14-8 FlashCopy consistency groups

If you use `db2acsutil` to query backup status, you can see similar output as in Example 14-28 here. Note that `acscim -D` queries the CIM agent on SVC MC to get the FlashCopy status periodically, so the Bytes completed and Bytes total are not updated in a timely manner. For a timely FlashCopy progress, you can check it from the SVC console as shown in Figure 14-7 on page 758.

*Example 14-28 db2acsutil query output*

```
$ db2acsutil query show details
      Instance : db2snap1
      Database : TEST
      Partition : 1
      Image timestamp : 20080501145123
      Host : Baltic
      Owner :
      DB2 Version : 9.5.1
      Creation time : Thu May 1 13:53:06 2008
      First active log (chain:file) : 0:0
      Metadata bytes : 6196
      Progress state : Successful
      Usability state : Remotely mountable + Repetitively restorable + Swap
      restorable + Physical protection + Full copy
      Bytes completed : 21474836480
      Bytes total : 21474836480

      Instance : db2snap1
      Database : TEST
      Partition : 0
```



```
Image timestamp : 20080501145123
Host : Banda
Owner :
DB2 Version : 9.5.1
Creation time : Thu May 1 14:52:16 2008
First active log (chain:file) : 0:0
Metadata bytes : 6196
Progress state : In progress
Usability state : Remotely mountable + Repetitively restorable + Swap
restorable + Physical protection + Full copy + Background_monitor pending
Bytes completed : 15676630630
Bytes total : 21474836480
```

---

Unless the Progress states on all database partitions are successful, you cannot use the backup image to restore the database. After FlashCopy finishes, DB2 ACS deletes the FlashCopy mappings and consistency groups automatically. For more information about the backup and restore cycle, refer to *Backup and Restore cycles* in Chapter 7 of the *TSM ACS V5.5 Installation and User's Guide*, SC33-8330, available at:

<http://publib.boulder.ibm.com/infocenter/tivihelp/v1r1/topic/com.ibm.itmfacs.doc/fbrd0000.pdf>

## 14.3 Recover database command

First introduced in DB2 V9.1, the **recover database** command consolidates a number of steps previously required with the **restore database** and **roll forward database** commands.

While retaining some very useful niche functionality, such as initializing an HADR standby database that needs to be left in rollforward pending state, or redirected restores on SMS or DMS table space containers, the often time-consuming and effort-intensive pairing of **restore database** and **roll forward database** commands has become unnecessary for common situations.

### Feature summary

As a feature we believe to be most beneficial for common recovery scenarios, the default behavior for the most simple **db2 recover database <dbname>** command syntax is for DB2 to use the most recent backup file for that database in the recovery history file and then perform rollforward recovery to the end of logs. Not only that, but even with a recovery to point in time (PIT), DB2 9.5 automatically chooses the appropriate backup file and rolls logs forward from the appropriate log chain to that PIT. It is no longer necessary to specify a timestamp and physical location to choose the correct backup file as well as the timestamp to roll forward to.

A quick look through the DB2 V9.1 or V9.5 *Command Reference*, SC23-5846 or Information Center highlights the features of the **recover database** command:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0011703.htm>

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0011703.htm>

This new command has parameters that allow us to specify the following items:

- ▶ Point In Time (PIT) recovery to *<isotime>* (UTC or local).
- ▶ Explicit naming of a recovery history file for partitioned databases.
- ▶ Various configurations of single or multiple partition recovery. The recover database command for partitioned databases must be issued from the catalog node.
- ▶ Custom decompression libraries and options.
- ▶ Restart of the **recover database** command, or optionally, using **restore database** or **rollforward database** commands to more efficiently continue a **recover database** process at certain stages after any failures. This latter feature avoids lengthy waits for restart logic to find the appropriate place to commence.

Judging from the foregoing list of features, the **recover database** command appears to be trying for a balance between simplicity and versatility.

## Feature testing

To give you an indication of the results to expect from usage of the **recover database** command, we proceed to run the command with a variety of parameters, on both non-partitioned and partitioned database test-beds.

### ***Preparation for single-partitioned database recovery examples***

We use DB2 9.5 Data Server installed on SUSE Linux Enterprise Server 10.1 x64 for our testing.

We first create a plain empty database, configure it to use archive logging, and create a default SMS table space inside it, performing updates and backups to prepare for recovery testing. The only prerequisite was that our test-bed database have archive logging enabled.

We run the bash scripts shown in Example 14-29. Script 1 contains a create table and insert statement, to set up a test table and populate it with an initial row. Script 2 contains an embedded loop (255 x 255 iterations). The inner loop performs 255 update statements on our test table to create uniquely identifiable table row values, as well as transaction log records.

The outer loop performs **backup database** commands to force 255 archive logs out. This is designed to give us a fair range of PIT recovery choices spread out over a decent period of time. These same scripts are also used for the testing that we carry out in 14.4, “Recovery object management” on page 791. We save the console output of our script as a guide for confirming successful results of a recover to PIT.

---

*Example 14-29 Scripts to create test-bed for recover database*

---

*Script 1:*

```
#!/bin/bash
db2 connect to test
db2 "create table testtab (col1 int not null, col2 char(20), col3 timestamp )"
db2 "insert into testtab values (1,'Testrow',CURRENT_TIMESTAMP)"
```

*Script 2:*

```
#!/bin/bash
c1=1
while [ $c1 -le 255 ] ; do
  db2 connect to test
  c2=1
  while [ $c2 -le 255 ] ; do
    (( c2++ ))
    db2 "update testtab set col2='Testtab_"$c1_"$c2"', col3=CURRENT_TIMESTAMP
where col1=1" > crapout.txt
  done
  db2 "select * from testtab"
  db2 backup db test online to /db2/backup compress include logs
  echo "Updated val: "$c1_"$c2
  (( c1++ ))
done
```

---

### ***Preparation for multi-partitioned database recovery examples***

Our second test-bed consists of a very basic partitioned database environment, where we reproduce the same data updates as the first test-bed. We perform similar recovery testing. To prepare for testing of the **recover database** command on a multi partition database with two logical partitions and two default partition groups (IBMCATGROUP, IBMDEFAULTGROUP), we carry out the following steps on our test servers (host name: lead, instance: db2inst1, database: test).

1. Configure DB2 to use SSH:

We ensure that we have password-less Secure Shell (SSH) in place for intra-partition communication, and configure DB2 to use SSH rather than Remote Shell (rsh). These steps are explained in detail in the preparation subsection of Chapter 11.3, “Automating HADR takeover with TSA on AIX” on page 377.

As a summary, we simply append our generated SSH public key files `/home/db2inst1/.ssh/dsa.pub` and `rsa.pub` to the `/home/db2inst1/.ssh/authorized_keys` file, then issue the command **db2set DB2RSHCMD=/usr/bin/ssh** and recycle the DB2 instance.

2. Create an instance:

```
db2icrt -p 55000 -u db2fenc1 db2inst1
```

3. Confirm that certain kernel parameters are set and stored. As the root user we perform **vi /etc/sysctl.conf** to add the following lines where necessary:

```
kernel.shmmax=768000000
kernel.shmmni=4096
kernel.sem=250 32000 100 128
```

Note that the values shown are basic, and might need to be adjusted for your environment.

These values are made effective with the following command:

```
lead:~ #sysctl -p
```

4. Configure DB2 with two logical partitions, both on a single server. As the DB2 instance user, we edit `/home/db2inst1/sqllib/db2nodes.cfg` to have the following entries:

```
0 lead 0
1 lead 1
```

5. Check `/etc/services` to ensure that DB2 Fast Communication Manager (FCM) ports are reserved for intra-partition communications in the DB2 instance. Our environment shown in Example 14-30 is using ports 60000 to 60003 for FCM, and 55000 for external communications on db2inst1:

*Example 14-30 /etc/services DB2 ports for intra-partition communication*

---

```
lead:~ #cat /etc/services |grep -i db2
db2inst1i      55000/tcp
DB2_db2inst1  60000/tcp
DB2_db2inst1_1 60001/tcp
DB2_db2inst1_2 60002/tcp
DB2_db2inst1_END 60003/tcp
```

---

6. Create database

We have the DB2 instance user issue **db2start**, and create a database named TEST. With all the above prerequisite tasks carried out, the **create database** command also adds two default partition groups. In Example 14-31 we list the partitions for verification after connecting to the database.

### Example 14-31 Creating the partitioned database

---

```
db2inst1@lead:~> db2start
05/01/2008 03:36:35    0  0  SQL1063N  DB2START processing was successful.
05/01/2008 03:36:37    1  0  SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
db2inst1@lead:~> db2 create database test
DB20000I  The CREATE DATABASE command completed successfully.
db2inst1@lead:~> db2 connect to test
```

#### Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
db2inst1@lead:~> db2 list db partition groups
```

#### DATABASE PARTITION GROUP

```
-----
IBMCACTGROUP
IBMDEFAULTGROUP
```

---

7. Update the `NUM_DB_BACKUPS` database configuration (db cfg) parameter, so that the recovery history file keeps information for more than the default of 12 backup files:

```
db2inst1@lead:/db2> db2 get db cfg for test |grep NUM_DB
Number of database backups to retain (NUM_DB_BACKUPS) = 366
```

8. Prepare each partition of the database for archive logging. We only have to update the db cfg parameter `LOGARCHMETH1` once. After the required database deactivation, the offline backup which puts the archive logging into effect must be performed on all database partitions.

Example 14-32 shows how we use the extremely useful Single System View (SSV) feature of the **backup database** command from DB2 9.5, to simplify the backup of multiple database partitions. We explain SSV in detail in 14.1, “Single system view backup” on page 714, and this feature is also covered in the DB2 9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.1uw.wn.doc/doc/c0051390.html>

### Example 14-32 Initial backup database commands using SSV feature

---

```
db2inst1@lead:~> db2 connect reset
DB20000I  The SQL command completed successfully.
db2inst1@lead:~> db2 update db cfg for test using logarchmeth1 disk:/db2/archlog
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lead:~> db2 deactivate db test
```

```

SQL1496W Deactivate database is successful, but the database was not
activated.
db2inst1@lead:~> db2 backup db test on all dbpartitionnums to /db2/backup compress
Part Result
-----
0001 DB20000I The BACKUP DATABASE command completed successfully.
0001 DB20000I The BACKUP DATABASE command completed successfully.

Backup successful. The timestamp for this backup image is : 20080501061902

```

---

After the offline database backup, our test database is in archive logging mode, and we can test that our **backup database test on all dbpartitionnums online** command works as shown Example 14-33.

*Example 14-33 Online database backup with SSV*

```

db2inst1@lead:~> db2 backup db test on all dbpartitionnums online to
/db2/backup compress include logs
Part Result
-----
0000 DB20000I The BACKUP DATABASE command completed successfully.
0001 DB20000I The BACKUP DATABASE command completed successfully.

Backup successful. The timestamp for this backup image is : 20080501061910

```

---

While there is no change to our test-bed Script 1, we do adjust the backup command in our Script 2 slightly for our partitioned database, using the new DB2 9.5 **backup database** parameter **on all dbpartitionnums**. We reduce the iteration limit in the outer loop from 255 to 127, partly due to an elapsed time overhead in carrying out partitioned database backups as opposed to standard database backups, and partly due to a limitation of free disk space on our test-bed platform, which makes the information in 14.4, “Recovery object management” on page 791 all the more relevant and useful. Our customized Script 2 for partitioned database test-bed is shown in Example 14-34.

*Example 14-34 Test-bed preparation for partitioned database testing*

```

#!/bin/bash
c1=1
while [ $c1 -le 127 ] ; do
  db2 connect to test
  c2=1
  while [ $c2 -le 255 ] ; do
    (( c2++ ))
    db2 "update testtab set col2='Testtab_"$c1_"_"$c2"', col3=CURRENT_TIMESTAMP
where col1=1" > crapout.txt
  done
  db2 "select * from testtab"

```

```
db2 backup db test on all dbpartitionnums online to /db2/backup compress include
logs
echo "Updated val: "$c1_"_$c2
(( c1++ ))
done
```

---

### ***Protecting log files***

The first thing any DB2 administrator must remember to do in a PIT recovery scenario is to perform a backup. Specifically, this means performing a manual backup of all relevant active and archive log files. If there is no evident database corruption, a full offline backup of the current database is also often performed beforehand. These actions are important for administrators because the very act of recovery in DB2 starts a new log chain that would truncate and reuse the existing log file names.

While DB2 knows which version of which physical log file sharing the same name belongs to which given log chain, it is much more difficult for an administrator to manually keep track of this during a major recovery event. If the database has been deactivated and a full database backup is performed beforehand, whether online or offline, active log files are safely truncated and archived into the current log chain. Thus, administrators know that everything required to get the database back to the current state is safe and in one log chain, before they commence what could become a series of recoveries to various points in time (creating many confusing log chains in the process) to satisfy the requirements of users.

While we can always successfully recover to points in time progressively further back using the one set of log files from the original log chain, we cannot just start recovering again to a PIT later than we have already recovered to without ensuring that we have those log records from the recovery chain chronologically prior to initial recovery activities. Generally speaking, the fact that DB2 saves the old recovery chain of archive log files in a separate subdirectory means they are relatively safe. Log file names with numbering later than the recovery PIT can get reused. However, this also means that multiple versions of the same file name can exist. In DB2 V9.5, log files required for PIT recovery are packaged along with the database backup image. For more information about log file management, refer to the Information Center:

<http://publib.boulder.ibm.com/infocenter/db21uw/v9r5/topic/com.ibm.db2.1uw.admin.ha.doc/doc/c0006085.html>

The recovery history file, which DB2 uses as an index to search for appropriate backup files and log records, is automatically backed up inside the backup file. However, recovery of an old version of an existing database just uses the current active recovery history file by default, which contains information on log files later than those that the old database backup file knows about.

The recovery history file itself is explained in the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006136.html>

The improvements for simplicity and reliability of recovery have been made over DB2 V9.1 to DB2 9.5. For instance, the log control file, which DB2 uses to perform automatic crash recovery at the first connection attempt after a database failure, is now being written to two copies (SQLOGCTL.LFH.1 and SQLOGCTL.LFH.2) rather than one (SQLOGCTL.LFH). Information on log control files can be found here:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0051340.html>

Log files that support rollforward recovery to a stable PIT just after an online backup can optionally be added to a backup file with the INCLUDE LOGS backup command parameter. Prior to DB2 9.5, this was not even an option for partitioned database backups, which also had to be backed up individually. We use the INCLUDE LOGS parameter in our scripts, but we generally only require logs included with online database backup images for restoring a database to the arbitrary PIT when the backup was taken. While not necessarily a desirable PIT, it is adequate for creating a new clone database on any server of the same platform type, from standalone online backup files. Otherwise, cloning to a new database would be done by restoring a full offline backup file to a new database name, which does not require any associated log files, nor does it require any **rollforward database** command.

Cloning a database to extract old data without overwriting the existing live database does not always allow the luxury of using an offline database backup file. While this section deals primarily with the **recover database** command, we direct your attention to a closely related new feature in V9.5 that allows for the **rollforward database** command to specify TO END OF BACKUP as a parameter. This feature is extremely useful for both non-partitioned and partitioned database backups, to automatically find the earliest consistent PIT where a rollforward can complete and make the database available for use. This is effectively the minimum rollforward PIT, with the maximum being “to end of logs”. Further information on this feature can be found in the DB2 V9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.wn.doc/doc/c0052796.html>

For our test environment, we perform a **backup database** command so we can restart recovery to PIT from our current state if necessary. At present, our database is deactivated, because there are no connections to it, and we have more than enough full backups of our database, including one of the current state of the data in our table, so we are safe to start testing. We achieve a



backup of our relevant log files by first finding where the active logs and archive logs actually are. Interrogation of the **get db cfg** output shows both the active and archive log locations (at least, the base subdirectory for archive logs):

```
db2inst0@lead:~>db2 get db cfg for test |egrep "Path|LOGARCHMETH"
Path to log files= /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/
First log archive method          (LOGARCHMETH1) = DISK:/db2/archlog/
Second log archive method         (LOGARCHMETH2) = OFF
```

Exploring the /db2/archlog base subdirectory, we find that the most recent chain of archive logs has ended up in:

```
/db2/archlog/db2inst0/TEST/NODE0000/C0000002>
```

Previous log chains are in the C0000000 and C0000001 subdirectories, from earlier activities on our test database.

We back up the active and archive log files with a few simple commands, listed in Example 14-35.

*Example 14-35 Back up log files*

---

```
db2 connect reset
db2 deactivate db test
cd /db2/archlog
mkdir -p backup
cd backup
cp -R /db2/archlog/db2inst0/TEST/NODE0000/C0000002 .
cp -R /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR .
```

---

Now that our log files are all safe, we can concentrate on looking at the files and timestamp ranges we want to deal with, summarized in Example 14-36.

*Example 14-36 First and last backup, log and table values in recovery range*

---

```
Backups are in: /db2/backup
First backup: TEST.0.db2inst0.NODE0000.CATN0000.20080424180908.001
Last backup: TEST.0.db2inst0.NODE0000.CATN0000.20080424183523.001

Archive logs are in: /db2/archlog/db2inst0/TEST/NODE0000/C0000002
First archive log: 2008-04-24 18:03 S0000015.LOG
Last archive log: 2008-04-24 18:35 S0000270.LOG

Active logs are in: /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR
First active log: 2008-04-25 10:53 S0000271.LOG
Last active log: 2008-04-25 10:53 S0000283.LOG

First timestamp in table at Testtab_1_256: 2008-04-24-18.09.07.974500
Last timestamp in table at Testtab_255_256: 2008-04-24-18.35.23.121000
```

---

### ***Compressed database backup files***

Something to note at this point is that our backup files are all compressed, and the **recover database** command is capable of decompressing without our needing to specify any compression library parameter. This would not be the case if we compressed the backup using a custom library. More information on custom compression libraries and the DB2 Compression API can be found here:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.apdv.api.doc/doc/r0011163.html>

We have a vanilla un-compressed backup of our test database sitting around the 100 MB mark: 100708352 bytes. DB2's compressed backup of that same database gets: 20467712 bytes, or about 20% of the original size in this particular case. Though compression reduces the backup file size, it can take more time in backup.

### ***Test expectations and checking methodology***

What we hope to see after each recovery test is some indication of whether we have retrieved the expected data from the recovery we have requested.

We have designed our test table to have a numbering scheme in the second column (col2) that we can use to deduce from the output of our script whether we are getting the correct results from any recover to PIT. Example 14-37 shows some output from our script to indicate where we are matching column values against the backup file's timestamp.

#### *Example 14-37 Extract of table update and database backup script output*

---

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TEST
```

```
COL1      COL2      COL3
-----
          1 Testtab_128_256      2008-04-24-18.22.01.971900
```

1 record(s) selected.

Backup successful. The timestamp for this backup image is : 20080424182202

Updated val: 128\_256

---

We supplement our numbering scheme in column col2 with a timestamp column col3. Our script updates our timestamp column col3 with the `CURRENT_TIMESTAMP` special register, and effectively issues an implicit

commit. While this value can never match the exact timestamp of a subsequent backup, it should be useful as a guide when we perform our recover to PIT, to see what kind of granularity DB2 limits us to in terms of rolling forward to specific log records as a part of the recover database command. Technically, the results of this testing would be no different than if we manually issued a restore database followed by a rollforward database to the same PIT timestamp. The recovery database command just makes it much easier to achieve.

It is important not to rely on a single form of tracking mechanism for our tests, which is why we employ both a timestamp column (col3), and an arbitrary counter value column (col2). In the real world, users who want old data matching a given value consider recovery to be a success if they get data rows containing that given value. They would consider recovery to be a failure even if the database is recovered successfully to the exact PIT the user requests, but it gives them unexpected data. On the other hand, users who track their data solely by timestamp, most likely know exactly what PIT they want to recover to, but might not know what values the data would have as a result.

Before performing any tests, we check the current value of the data in the test table with a DB2 select statement as shown in Example 14-38.

*Example 14-38 Current content of our test table*

---

```
db2inst0@lead:~> db2 "select * from testtab"
```

COL1	COL2	COL3
-----	-----	-----
	1 Testtab_255_256	2008-04-24-18.35.23.271220

1 record(s) selected.

---

This tells us our current data. If we recover using defaults, rolling forward to the end of our logs, we would expect the same output from that select statement.

### ***Test 1: Using recover database command with default values***

Our first test is to simply use the recover database command using defaults, that is, with no parameters other than the database name, as shown in Example 14-39. As explained in “Feature summary” on page 759, we expect to see DB2 simply applying the most recent full backup and rolling forward to end of logs. This test does not show any data being changed, because the syntax that we are using is designed primarily for the purpose of rebuilding a broken or corrupt database to the point in time of the latest committed transaction.

*Example 14-39 First recover database test - simple, no parameters*

---

```
db2inst0@lead:~> db2 connect reset
DB20000I The SQL command completed successfully.
```

```
db2inst0@lead:~> db2 deactivate db test
DB20000I The DEACTIVATE DATABASE command completed successfully.
db2inst0@lead:~> db2 recover db test
```

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            = S0000270.LOG - S0000271.LOG
Last committed transaction     = 2008-04-25-08.50.58.000000 Local
```

```
DB20000I The RECOVER DATABASE command completed successfully.
db2inst0@lead:~> db2 connect to test
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TEST
```

```
db2inst0@lead:~> db2 "select * from testtab"
```

COL1	COL2	COL3
1	Testtab_255_256	2008-04-24-18.35.23.271220

```
1 record(s) selected.
```

The result is just as expected: The data in our test table row is just as we left it before executing the **recover database** command. Because we ran this **recover database** command after running a **backup database** command the next morning after the test run, the last committed transaction is shown as a later timestamp (2008-04-25-08.50.58.000000), than the COL3 value (2008-04-24-18.35.23.271220). Our result remains the same in the table, because there have been no updates to the data since our script run ended at the timestamp shown in the COL3 value.

To clarify some particulars in the DB2 Information Center, we noticed that the **recover database** command description states that a connection to the database is required for recovery of an existing database, so we did try connecting to the database and attempting two forms of the **recover database** command with no parameters at all. We were checking whether DB2 might perform an implicit recover using the current database connection, potentially allowing an even shorter command syntax.

As expected, we received the messages in Example 14-40. This is because DB2 automatically creates the database connection inside the utility, and still requires that we specify which database we want to recover.

*Example 14-40 DB2 recover database command requires explicit database name*

---

```
db2inst0@lead:~> db2 recover db
SQL0104N  An unexpected token "END-OF-STATEMENT" was found following "DB".
Expected tokens may include: "<identifier>".  SQLSTATE=42601

db2inst0@lead:~> db2 recover database
SQL0104N  An unexpected token "END-OF-STATEMENT" was found following
"DATABASE".  Expected tokens may include: "<identifier>".  SQLSTATE=42601
```

---

We performed yet another test of the **recover database** command with only the database name as a parameter, after explicitly connecting to the database, and that worked. Attempting the **recover database** while another user was connected to that database caused the recover command to fail with the message:

```
SQL1035N  The database is currently in use.  SQLSTATE=57019
```

This message is just another safety mechanism to help avoid unnecessary data loss (protecting users from themselves and each other), although if we are going to perform a recovery anyway, all user and application connectivity to the DB2 database should have been disabled as a primary data integrity safety measure.

We also found that after completion of the recover database command, DB2 releases the database connection made during execution of that command. Any subsequent SQL statements issued require another explicit database connection.

### ***Test 2: PIT recovery***

Our next test is to recover to a PIT. We aim for a mid-point in our test run, namely, when the table contained data in column **col12** matching **Testtab\_128\_256**. Because we know how the timing of our updates relates to our backup database command, we choose a PIT matching the timestamp embedded in the database backup filename itself, the same timestamp which appears in the output from our script.

Normally, in an environment where we do not have a good idea of exactly what PIT to recover to, a DBA has to take cues from application support or users, who themselves might only have a vague idea, and might need to retry several times to get the exact PIT desired. In some cases, recovery is done by means of cloning an old version of the live production database from the backup file into a new database to avoid impacting the live environment.

In this case, the backup file's timestamp matching the Testtab\_128\_256 column value is 20080424182202. The ISOTIME parameter in the **recover database** command has to match this syntax: yyyy-mm-dd-hh.mm.ss.nnnnnn (year, month, day, hour, minutes, seconds, microseconds). Converting our timestamp into the expected format means that we use the value 2008-04-24-18.22.02.

The results of our first attempt are shown in Example 14-41.

*Example 14-41 Recovery to PIT - our initial attempt*

---

```
db2inst0@lead:~> db2 recover database test to 2008-04-24-18.22.02
SQL2163N  DB2 is unable to locate a backup image in the Recovery History File
to recover the database to the given point in time on database partition
number "0".
```

---

Here we run into an interesting problem: Our recovery history file only contains 15 references to the most recent backup and restore database commands, as we illustrate in Example 14-42. As a result, DB2 cannot determine which log files would work with the unknown backup file. Two of the 15 references are to restore commands, and one is to an un-compressed offline backup we made after running the main test-bed generation script. The earliest PIT we could restore and roll forward from was 20080424183414001, which is very close to the end of our test backups, and therefore rather useless.

*Example 14-42 Trouble ahead - no relevant recovery history file entries*

---

```
db2inst0@lead:/db2/backup> db2 list history backup all for db test
```

List History File for test

Number of matching file entries = 15

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
```

```
-----
B D 20080424183414001 N D S0000259.LOG S0000259.LOG
-----
```

Contains 3 tablespace(s):

```
00001 SYSCATSPACE
00002 USERSPACE1
00003 SYSTOOLSPACE
```

```
-----
Comment: DB2 BACKUP TEST ONLINE
Start Time: 20080424183414
End Time: 20080424183416
Status: E
-----
```

```
EID: 520 Location: /db2/backup
```

```
...
```

---

The default db cfg parameter setting for the recovery history file is to only retain indexing information on the last 12 backup files, or the last 366 days of activity. DB2 removes entries if they exceed either setting. Our default database uses the defaults, as shown in Example 14-43.

*Example 14-43 Two very relevant output lines from the db2 get db cfg command*

---

```
Number of database backups to retain (NUM_DB_BACKUPS) = 12
Recovery history retention (days) (REC_HIS_RETENTN) = 366
```

---

**Note:** This does not make recovery to an earlier PIT impossible, just tedious. Basically, if DB2 has no recovery history records in order to look up old database backup and archive log files, then restoring any one of those old database backups only allows us to restore to the time the backup was taken. Even specifying the **replace history file** parameter in the restore database command, we cannot roll forward beyond the time that database backup was taken, because the recovery history file contained in each database backup does not contain references to log records written after the backup occurred.

In our test environment, references to only the 12 most recent backup and archive files are being kept in the recovery history file, while all the older backup and archive log files still remain on disk. If we were to consider a hypothetical PIT recovery scenario to an un-referenced timestamp, we would perform a staged restore: First, issue a **restore database ... replace history file** command, specifying a backup file with recovery history within 12 backup images of the one we want. Then we issue our original **recover database** command specifying the desired timestamp, as the replaced recovery history file would then contain the information so DB2 knows which backup file and archive log files to use.

After updating the NUM\_DB\_BACKUPS database configuration parameter with commands in Example 14-44, we execute our test script 2 again. This script repopulates our recovery history file with backup and log files, and updates the table with reference data to compare after performing recovery.

*Example 14-44 Making our recovery history file remember more than 12 backups*

---

```
db2inst0@lead:~> db2 update db cfg for test using NUM_DB_BACKUPS 512
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst0@lead:~> db2 deactivate db test
SQL1496W Deactivate database is successful, but the database was not
activated.
```

---

To confirm that we the required backups are still residing in the recovery history file, we issue a **list history** command. Example 14-45 shows the command and first entry of the output, proving that we do indeed have entries going back well before the ones we require for our test:

*Example 14-45 List History command and initial output*

---

```
db2inst0@lead:~> db2 list history backup all for db test
```

```
List History File for test
```

```
Number of matching file entries = 269
```

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
B D 20080424183420001 N D S0000260.LOG S0000260.LOG
-----
```

```
Contains 3 tablespace(s):
```

```
00001 SYSCATSPACE
00002 USERSPACE1
00003 SYSTOOLSPACE
-----
```

```
Comment: DB2 BACKUP TEST ONLINE
Start Time: 20080424183420
End Time: 20080424183422
Status: A
-----
```

```
EID: 522 Location: /db2/backup
```

---

The details of backups, logs, and timestamps for our second test-bed run are shown in Example 14-46. We have made backups of the new logs.

*Example 14-46 DB2 backup, log, timestamp ranges from second test-bed*

---

```
Backups are in: /db2/backup
```

```
First backup: TEST.0.db2inst0.NODE0000.CATN0000.20080425164919.001
```

```
Last backup: TEST.0.db2inst0.NODE0000.CATN0000.20080425194602.001
```

```
Every recover database command starts a new log chain.
```

```
Archive logs are in: /db2/archlog/db2inst0/TEST/NODE0000/C0000004
```

```
First archive log: 2008-04-28 11:19 S0000526.LOG
```

```
Last archive log: 2008-04-28 11:19 S0000272.LOG
```

```
Active logs are in: /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR
```

```
First active log: 2008-04-28 10:08 S0000527.LOG
```

```
Last active log: 2008-04-28 10:08 S0000539.LOG
```

DB2 also lists S0000527.LOG as the first active log file in output from a **get db cfg for test** command.

```
First timestamp in table at Testtab_1_256: 2008-04-25-19.19.43.725000
```

```
Last timestamp in table at Testtab_255_256: 2008-04-25-19.46.02.271220
```

---



Looking at the output from our script at the same mid-point of the test run in Example 14-47, we establish that we want to recover to a PIT just after this row was created; that is, after the commit point occurred to update that value in the table. The output is from the `select *` from `testtab` SQL in the outer loop in our script 2 and the backup database statement.

*Example 14-47 Output extract from the mid-point of our test script log*

---

```
COL1          COL2          COL3
-----
1 Testtab_128_256  2008-04-25-19.32.33.971900

1 record(s) selected.
```

Backup successful. The timestamp for this backup image is : 20080425193234

---

For our purposes, the **backup database** timestamp should be granular enough as a specification to the correct second: 20080425193234. After converting that value to a compatible timestamp format, our **recover database** command is executed as shown in Example 14-48.

*Example 14-48 Second try: Recover database to PIT*

---

```
db2inst0@lead:~> db2 recover db test to 2008-04-25-19.32.34
```

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           = S0000398.LOG - S0000400.LOG
Last committed transaction    = 2008-04-25-19.32.34.000000 Local
```

```
DB20000I The RECOVER DATABASE command completed successfully.
```

```
db2inst0@lead:~> db2 connect to test
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TEST
```

```
db2inst0@lead:~> db2 "select * from testtab"
```

---

```
COL1          COL2          COL3
-----
```

1 Testtab\_128\_256 2008-04-25-19.32.33.971900

1 record(s) selected.

---

We have exactly the output we were hoping for, the value of col2 as at the midpoint of our tests: Testtab\_128\_256.

### **Test 2: Behind the scenes**

In an attempt to discover what is going on behind the scenes of our recovery, we now examine the background workings of the **recover database** command just issued from the DB2 notify log and db2diag.log.

### **Examining db2diag.log**

The first action DB2 takes is to perform a log truncation and archive of the current active log (S0000527), shown in Example 14-49.

#### *Example 14-49 db2inst0.nfy tail extract from recover activity part 1 of 4*

---

```
2008-04-28-12.11.29.867045 Instance:db2inst0 Node:000
PID:15199(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgArchiveLogFile Probe:3108
```

```
ADM1844I Started archive for log file "S0000527.LOG".
```

```
2008-04-28-12.11.29.928046 Instance:db2inst0 Node:000
PID:15199(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgArchiveLogFile Probe:3170
```

```
ADM1846I Completed archive for log file "S0000527.LOG" to
"/db2/archlog/db2inst0/TEST/NODE0000/C0000004/" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".
```

---

No mention is made of the backup file being used, instead, the DB2 notify log simply tracks the requests for log files being used for the rollforward recovery phase, shown in Example 14-50.

#### *Example 14-50 db2inst0.nfy tail extract from recover activity part 2 of 4*

---

```
2008-04-28-12.11.35.948517 Instance:db2inst0 Node:000
PID:26911(db2agent (TEST) 0) TID:2169317664 Appid:*LOCAL.db2inst0.080428191137
data protection services sqlufrol Probe:980 Database:TEST
```

```
ADM1602W Rollforward recovery has been initiated.
```

```
2008-04-28-12.11.36.065754 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4130
```

```
ADM1843I Started retrieve for log file "S0000398.LOG".
```

```
2008-04-28-12.11.36.081216 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4148

ADM1845I Completed retrieve for log file "S0000398.LOG" on chain "4" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".

2008-04-28-12.11.36.092244 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4130

ADM1843I Started retrieve for log file "S0000399.LOG".

2008-04-28-12.11.36.112739 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4148

ADM1845I Completed retrieve for log file "S0000399.LOG" on chain "4" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".

2008-04-28-12.11.36.118151 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4130

ADM1843I Started retrieve for log file "S0000400.LOG".

2008-04-28-12.11.36.126914 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4148

ADM1845I Completed retrieve for log file "S0000400.LOG" on chain "4" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".

2008-04-28-12.11.36.132520 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4130

ADM1843I Started retrieve for log file "S0000401.LOG".

2008-04-28-12.11.36.136606 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgRetrieveLogFile Probe:4148

ADM1845I Completed retrieve for log file "S0000401.LOG" on chain "4" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".
```

---

We see in Example 14-50 that the log file S0000401.LOG was requested and retrieved, despite only S0000398 to S0000400 being listed as used by the output of the actual recover database command. It is most likely that log S0000401 was retrieved for checking, but did not contain any entries for dependant objects prior to the commit point used to match our PIT parameter.

Example 14-51 shows DB2 completing the rollforward recovery phase.

*Example 14-51 db2inst0.nfy tail extract from recover activity part 3 of 4*

---

```
2008-04-28-12.11.37.530821 Instance:db2inst0 Node:000
PID:26911(db2agent (TEST) 0) TID:2169317664 Appid:*LOCAL.db2inst0.080428191137
data protection services sqlufrol Probe:8180 Database:TEST
```

```
ADM1611W The rollforward recovery phase has been completed.
```

---

As seen in Example 14-52, after completing the recover database, DB2 truncates the log using the numbering as of the PIT recovered to, and then archives this and subsequent logs to a new log chain in a new subdirectory, keeping track of both log chains in the recovery history file. This goes some way to protecting old archive logs from previous recovery points. However, it is no substitute for backing up all logs prior to commencing multiple recover commands. An initial back up in this case avoids confusion about which S0000400.LOG contains non-truncated good data for subsequent recover commands to later points in time.

*Example 14-52 db2inst0.nfy tail extract from recover activity part 4 of 4*

---

```
2008-04-28-12.11.37.540302 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgArchiveLogFile Probe:3108
```

```
ADM1844I Started archive for log file "S0000400.LOG".
```

```
2008-04-28-12.11.37.566758 Instance:db2inst0 Node:000
PID:30017(db2logmgr (TEST) 0) TID:2169317664 Appid:none
data protection services sqlpgArchiveLogFile Probe:3170
```

```
ADM1846I Completed archive for log file "S0000400.LOG" to
"/db2/archlog/db2inst0/TEST/NODE0000/C0000005/" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/".
```

---

DB2 log files are also truncated when there are no more active connections to the database, and a new log file is started when an **activate database** or **connect** command is issued.

### Examining the db2diag.log

Example 14-53 starts off our second series of examples, to examine a representative summary of the db2diag.log tail output from the recover database command. The full output, even from such a small and simple recover to PIT, is extremely detailed and we only need to examine key aspects of it.

Initially, we can see the same activity here as shown in the DB2 notify log, truncating and archiving S0000527.LOG.

*Example 14-53 DB2 db2diag.log tail extract from recover activity part 1 of 9*

---

```
2008-04-28-12.11.29.841464-420 I2000468E343      LEVEL: Info
PID      : 15200                TID : 47354183756064PROC : db2loggr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpghck, probe:1390
DATA #1 : <preformatted>
ExtNum 527 state 101 baselsn 0000000084BA0000 nextlsn 0000000084BC0BC8

2008-04-28-12.11.29.880673-420 E2000812E327      LEVEL: Warning
PID      : 15199                TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3108
MESSAGE : ADM18441 Started archive for log file "S0000527.LOG".

2008-04-28-12.11.29.928192-420 E2001140E456      LEVEL: Warning
PID      : 15199                TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3170
MESSAGE : ADM18461 Completed archive for log file "S0000527.LOG" to
         "/db2/archlog/db2inst0/TEST/NODE0000/C0000004/" from
         "/db2/data/db2inst0/NODE0000/SQL00001/SQL0GDIR/".
```

---

In Example 14-54, DB2 tells us it is running a recover to PIT, and starts a new “restore sequence” to use for ongoing logging.

*Example 14-54 DB2 db2diag.log tail extract from recover activity part 2 of 9*

---

```
2008-04-28-12.11.33.570358-420 I2001597E439      LEVEL: Info
PID      : 2525                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1317              APPID: *LOCAL.db2inst0.080428191133
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlpCallRestore, probe:100
DATA #1 : <preformatted>
Running RECOVER to PIT 2008-04-26-02.32.34.000000 UTC.

2008-04-28-12.11.33.611867-420 I2002037E418      LEVEL: Info
PID      : 2525                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1317              APPID: *LOCAL.db2inst0.080428191133
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlpAddToRecoveryList, probe:5800
DATA #1 : <preformatted>
New restore seq at EID 545

2008-04-28-12.11.33.612173-420 I2002456E449      LEVEL: Warning
PID      : 2525                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1317              APPID: *LOCAL.db2inst0.080428191133
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlpAddToRecoveryList, probe:5420
MESSAGE : Ignore history entry with EID 546 for log 270 pLastEntry has EID 542.

2008-04-28-12.11.33.612797-420 I2002906E418      LEVEL: Info
PID      : 2525                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1317              APPID: *LOCAL.db2inst0.080428191133
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlpAddToRecoveryList, probe:5800
DATA #1 : <preformatted>
New restore seq at EID 550
```

---

Example 14-55 shows what we were looking for, the entry telling us which backup file is being used for the recovery.

*Example 14-55 DB2 db2diag.log tail extract from recover activity part 3 of 9*

---

```
2008-04-28-12.11.33.696031-420 I2003325E494 LEVEL: Warning
PID : 2525 TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0 NODE : 000 DB : TEST
APPHDL : 0-1317 APPID: *LOCAL.db2inst0.080428191133
AUTHID : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlpScanHistFileForRecoveryList, probe:3950
DATA #1 : <preformatted>
Selected backup timestamp: 20080425193228
Selected backup location: /db2/backup
Optype: N
```

---

This is primarily important to note because we specified a timestamp that appears to match the backup file after this one. We did not specify nanosecond granularity to exactly match that backup, so DB2 chose the previous backup file:

```
TEST.0.db2inst0.NODE0000.CATN0000.20080425193228.001
```

Rather than:

```
TEST.0.db2inst0.NODE0000.CATN0000.20080425193234.001
```

DB2 applies rollforward recovery after restoring that backup image, effectively until just before the backup file we used the timestamp from. The exceptions to this scenario would be either if the second backup file had a zero nanosecond timestamp, or if we had added an arbitrary small buffer value for our PIT timestamp to ensure that the backup file matching the timestamp we used gets chosen. For now, we see the restore process commencing in Example 14-56.

*Example 14-56 DB2 db2diag.log tail extract from recover activity part 4 of 9*

---

```
2008-04-28-12.11.33.789253-420 I2003820E316 LEVEL: Warning
PID : 29996 TID : 47354183756064PROC : db2med.2525.0 0
INSTANCE: db2inst0 NODE : 000
FUNCTION: DB2 UDB, database utilities, sqluMCRReadMediaHeader, probe:2110
MESSAGE : Transfer buffer size was changed from 0 to 3801088

2008-04-28-12.11.33.789675-420 E2004137E435 LEVEL: Info
PID : 2525 TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0 NODE : 000 DB : TEST
APPHDL : 0-1317 APPID: *LOCAL.db2inst0.080428191133
AUTHID : DB2INST0
FUNCTION: DB2 UDB, database utilities, sqludPrintStartingMsg, probe:1628
DATA #1 : <preformatted>
Starting a full database restore.
Agent PID: 2525
```

---

Skipping a number of db2diag.log entries, we eventually get a restore complete message in Example 14-57.

*Example 14-57 DB2 db2diag.log tail extract from recover activity part 5 of 9*

---

```
2008-04-28-12.11.35.557891-420 E2006317E374      LEVEL: Info
PID      : 2525                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0           NODE : 000                DB  : TEST
APPHDL   : 0-1317            APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INSTO
FUNCTION: DB2 UDB, database utilities, sqludrsa, probe:889
MESSAGE  : Restore Complete.
```

---

After skipping further messages, DB2 tells us in Example 14-58 that the rollforward recovery phase has been initiated.

*Example 14-58 DB2 db2diag.log tail extract from recover activity part 6 of 9*

---

```
2008-04-28-12.11.35.949082-420 E2007573E416      LEVEL: Warning
PID      : 26911              TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0           NODE : 000                DB  : TEST
APPHDL   : 0-1322            APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INSTO
FUNCTION: DB2 UDB, data protection services, sqlufrol, probe:980
MESSAGE  : ADM1602W Rollforward recovery has been initiated.
```

```
2008-04-28-12.11.35.949188-420 E2007990E459      LEVEL: Warning
PID      : 26911              TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0           NODE : 000                DB  : TEST
APPHDL   : 0-1322            APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INSTO
FUNCTION: DB2 UDB, data protection services, sqlufrol, probe:1210
MESSAGE  : ADM1603I DB2 is invoking the forward phase of the database
           rollforward recovery.
```

```
2008-04-28-12.11.35.949278-420 I2008450E534      LEVEL: Warning
PID      : 26911              TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0           NODE : 000                DB  : TEST
APPHDL   : 0-1322            APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INSTO
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
DATA #1 : <preformatted>
Invoking database rollforward forward recovery,
lowtranlsn 00000000647A620A in log file number 398
minbufflsn 00000000647A000C in log file number 398
```

---

More messages are skipped, up to our first log retrieval message in Example 14-59.

*Example 14-59 DB2 db2diag.log tail extract from recover activity part 7 of 9*

---

```
2008-04-28-12.11.36.063744-420 I2010692E442      LEVEL: Warning
PID      : 26911              TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0           NODE : 000                DB  : TEST
APPHDL   : 0-1322            APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INSTO
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
DATA #1 : <preformatted>
Using parallel recovery with 3 agents 5 QSets 15 queues and 6 chunks
```

```
2008-04-28-12.11.36.066862-420 E2011135E329      LEVEL: Warning
PID      : 30017              TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0           NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgRetrieveLogFile, probe:4130
MESSAGE  : ADM1843I Started retrieve for log file "S0000398.LOG".
```

```
2008-04-28-12.11.36.081345-420 E2011465E409      LEVEL: Warning
PID      : 30017                TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgRetrieveLogFile, probe:4148
MESSAGE  : ADM1845I Completed retrieve for log file "S0000398.LOG" on chain "4"
           from "/db2/data/db2inst0/NODE0000/SQL00001/SQL06DIR/".
```

---

Example 14-60 occurs after so many messages are skipped (mainly for log retrieval) until our forward log processing ends, and DB2 commences backward phase log processing.

*Example 14-60 DB2 db2diag.log tail extract from recover activity part 8 of 9*

---

```
2008-04-28-12.11.36.343914-420 I2014095E409      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL   : 0-1322              APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:1990
DATA #1  : <preformatted>
nextLsn 0000000064FA0052
```

```
2008-04-28-12.11.36.459047-420 E2014505E456      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL   : 0-1322              APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlufrol, probe:2120
MESSAGE  : ADM1605I DB2 is invoking the backward phase of database rollforward
           recovery.
```

```
2008-04-28-12.11.36.459235-420 I2014962E459      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL   : 0-1322              APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:2210
DATA #1  : <preformatted>
Invoking database rollforward backward recovery, nextLsn: 0000000064FA0052
```

```
2008-04-28-12.11.37.530953-420 E2015422E427      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL   : 0-1322              APPID: *LOCAL.db2inst0.080428191137
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, data protection services, sqlufrol, probe:8180
MESSAGE  : ADM1611W The rollforward recovery phase has been completed.
```

---

In Example 14-61, DB2 now truncates and archives the log file at the current PIT (which is back to S0000400.LOG) and starts a new log recovery chain.

*Example 14-61 DB2 db2diag.log tail extract from recover activity part 9 of 9*

---

```
2008-04-28-12.11.37.536946-420 I2015850E344      LEVEL: Info
PID      : 30018                TID : 47354183756064PROC : db2loggr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpghck, probe:1390
DATA #1  : <preformatted>
ExtNum 400 state 2101 baseLsn 0000000064FA0000 nextLsn 0000000064FA00B2
```

```
2008-04-28-12.11.37.540452-420 E2016195E327      LEVEL: Warning
```



```

PID      : 30017                TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3108
MESSAGE : ADM1844I Started archive for log file "S0000400.LOG".

2008-04-28-12.11.37.566832-420 E2016523E456      LEVEL: Warning
PID      : 30017                TID : 47354183756064PROC : db2logmgr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgArchiveLogFile, probe:3170
MESSAGE : ADM1846I Completed archive for log file "S0000400.LOG" to
"/db2/archlog/db2inst0/TEST/NODE0000/C0000005/" from
"/db2/data/db2inst0/NODE0000/SQL00001/SQL06DIR/".

2008-04-28-12.12.05.409066-420 E2016980E344      LEVEL: Info
PID      : 30841                TID : 47354183756064PROC : db2loggr (TEST) 0
INSTANCE: db2inst0            NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgLoggrInitDel01dLog, probe:1440
DATA #1 : <preformatted>
Cleaning up logs from RenameArchNum 400 to delLimit 401.

2008-04-28-12.12.05.791652-420 I2017325E527      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1321                APPID: *LOCAL.db2inst0.080428191205
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, PSM - SQL Procedure, psm_recover_all_procs, probe:10
MESSAGE : SQL procedure executables recovery has been initiated.
DATA #1 : Hexdump, 8 bytes
0x00007FFF321BA3E8 : 5445 5354 2020 2020                TEST

2008-04-28-12.12.05.792073-420 I2017853E516      LEVEL: Warning
PID      : 26911                TID : 47354183756064PROC : db2agent (TEST) 0
INSTANCE: db2inst0            NODE : 000      DB : TEST
APPHDL  : 0-1321                APPID: *LOCAL.db2inst0.080428191205
AUTHID   : DB2INST0
FUNCTION: DB2 UDB, PSM - SQL Procedure, psm_recover_all_procs, probe:900
MESSAGE : SQL procedure executables recovery ended. RC:
DATA #1 : Hexdump, 4 bytes
0x00007FFF321BA438 : 0000 0000                ....

```

---

### ***Test 3: Deciding the time for PIT recovery***

Here we attempt to establish two things, first, to what level of granularity we can restore to a PIT, and second, what we have to do to recover to a later PIT after performing a recovery that has overwritten and truncated log files and started a new recovery chain.

To answer the second question first, we simply check that the recovery history file still has a reference to the backup file timestamp range that we want. Because this is true in our scenario (with our current large NUM\_DB\_BACKUPS db cfg parameter), we can simply issue the recover database command with the different PIT. Because DB2 keeps track of non-unique log file names for different log sequences in the recovery history file, the entries that list log files from any given PIT back to the previous backup file should still be intact.

Now, to answer the first question, we attempt to recover to a PIT roughly in between two database backups, and see what result we get in our table data.

We are choosing a PIT between the two backup points immediately subsequent to our original recovery PIT, that is, to a point chronologically following the log truncation and new recovery chain created as a result of that previous recovery to PIT, such that we are using the logs from the previous recovery chain, not the new recovery chain. The corresponding backup files are:

```
TEST.0.db2inst0.NODE0000.CATN0000.20080425193234.001
TEST.0.db2inst0.NODE0000.CATN0000.20080425193239.001
```

The PIT we choose is: 2008-04-25-19.32.36.5, which is not exactly in the middle, but getting close, considering that we are not counting nanosecond timestamp values for those two backups. Because this PIT comes after our shell script table update statements start using the Testtab\_129\_1 to Testtab\_129\_255 value range, we are expecting to see a table entry with a col2 value somewhere around the middle of this range, coupled with (hopefully) a col3 value very close to what we expressed as a PIT.

The output from our **recover database** command is shown in Example 14-62.

*Example 14-62 The most granular recovery to PIT we are going to get*

---

```
db2inst0@lead: /> db2 recover db test to 2008-04-25-19.32.36.5
```

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           = S0000399.LOG - S0000400.LOG
Last committed transaction    = 2008-04-25-19.32.36.000000 Local
```

```
DB20000I The RECOVER DATABASE command completed successfully.
db2inst0@lead: /> db2 connect to test
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TEST
```

```
db2inst0@lead: /> db2 "select * from testtab"
```

```
COL1          COL2          COL3
-----
          1 Testtab_129_78    2008-04-25-19.32.37.008622
```

```
1 record(s) selected.
```

---

The results we get are expected. Although we can recover to a requested PIT, we cannot guarantee exactly to which commit timestamp DB2 rolls forward. In our next section, we attempt to address this with a technique using DB2's **quiesce tablespaces for table** command.

#### ***Test 4: Get quiesced for success***

In this test, we show how we can recover to a known stable point in time using DB2's **quiesce tablespaces for table** command. This technique is very useful, for example, when employed after OLTP ceases and before the batch runs commence. A quiesce at that time provides a viable stable recovery point without requiring an additional database backup. This allows batch processes to be rolled back if necessary, without risk of rolling back any valuable pre-batch transactions.

Note that DB2 registers entries for the **quiesce tablespaces for table** command in the recovery history file but not for the **quiesce database** and **quiesce instance** commands. The DB2 Information Center contains details on the **quiesce tablespaces for table** command, which acts in share, update, and exclusive mode:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002003.html>

To test that we can see a registered entry for a quiesce in the recovery history file, Example 14-63 shows how we connect to the database, issue the **quiesce tablespaces for table** command into exclusive mode, followed by a **list tablespaces** command to confirm the state as “Quiesced: EXCLUSIVE”.

#### *Example 14-63 Testing quiesce entry in recovery history file 1 of 3.*

---

```
db2inst0@lead: /> db2 connect to test
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.1.3
SQL authorization ID = DB2INST0
Local database alias = TEST
```

```
db2inst0@lead: /> db2 "quiesce tablespaces for table testtab exclusive"
DB20000I The QUIESCE TABLESPACES command completed successfully.
db2inst0@lead: /> db2 list tablespaces
```

Tablespaces for Current Database

*. irrelevant output removed here*

```
.
Tablespace ID      = 2
Name               = USERSPACE1
Type               = Database managed space
```

```

Contents                = All permanent data. Large table space.
State                   = 0x0004
  Detailed explanation:
    Quiesced: EXCLUSIVE

```

*. irrelevant output removed here*

...

---

We then issue a **quiesce tablespaces for table ... reset** command to take the table out of “Quiesced” mode in Example 14-64.

*Example 14-64 Testing quiesce entry in recovery history file 2 of 3.*

---

```

db2inst0@lead: /> db2 quiesce tablespaces for table testtab reset
DB20000I The QUIESCE TABLESPACES command completed successfully.
db2inst0@lead: /> db2 list tablespaces

```

Tablespaces for Current Database

*. irrelevant output removed here*

.

```

Tablespace ID          = 2
Name                   = USERSPACE1
Type                   = Database managed space
Contents               = All permanent data. Large table space.
State                  = 0x0000
  Detailed explanation:
    Normal

```

*. irrelevant output removed here*

...

---

Finally, a **list history** command shows us that both the quiesce and quiesce reset are registered in the recovery history file, each with a recovery PIT, shown in Example 14-65. We see three entries, the first entry’s Op column has a value of ‘X’ for a normal archive log. The Op column in the second and third entries has a value of ‘Q’ for the **quiesce** command. The Type column in the second entry has a value of ‘X’ for our **quiesce** command in **exclusive** mode, and for the third entry, the value of column Type is ‘Z’ for our **quiesce ... reset** command.

*Example 14-65 Testing quiesce entry in recovery history file 3 of 3.*

---

```

db2inst0@lead: /db2/backup> db2 list history since 20080428153500 for test

```

List History File for test

Number of matching file entries = 3

```

Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
X D 20080428153543 P D S0000401.LOG C0000006
-----

```

```

-----
Comment:
Start Time: 20080428153543
End Time:
Status: A
-----
EID: 1078 Location: /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR/S0000401.LOG
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
Q T 20080428153543 X S0000401.LOG S0000401.LOG
-----
"DB2INST0"."TESTTAB" resides in 1 tablespace(s):

00001 USERSPACE1
-----
Comment: 20080428223544GMT
Start Time: 20080428153543
End Time: 20080428153544
Status: A
-----
EID: 1079
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
Q T 20080428160953 Z S0000401.LOG S0000401.LOG
-----
"DB2INST0"."TESTTAB" resides in 1 tablespace(s):

00001 USERSPACE1
-----
Comment: 20080428230954GMT
Start Time: 20080428160953
End Time: 20080428160954
Status: A
-----
EID: 1080

```

Note that the primary purpose of the **quiesce tablespaces for table** command is to set the table space into a safe persistent state for the DB2 **load** command, which does not otherwise establish compatible S, U, or X lock states prior to execution, only *allow no access*, *allow read access*, or *lock with force*. The *allow no access* and *lock with force* options can force transactions to roll back and release locks on the target table.

Our technique for database-wide quiesce functionality can be achieved using the **quiesce tablespaces for table** command in exclusive mode, in conjunction with the **quiesce instance** command or the **quiesce database** command. This establishes a stable point in time for all table spaces in a given database. Our technique is not a perfect solution, and you should be aware of some restrictions:

- ▶ The **quiesce** command with instance or database scope should be run first to establish database-wide exclusive access, forcing off all users.
- ▶ A **quiesce database** cannot exclude access to the database from any user or group with DBADM or higher authority, and a **quiesce instance** cannot exclude access to the database or instance from any user or group with SYSMANT or higher authority.

- ▶ A system catalog table cannot be used with **quiesce tablespaces for table**.
- ▶ Timestamps for **quiesce tablespaces for table** commands registered in the recovery history file for all the table spaces in a given database do not match each other. If the timestamps all occur within a period of exclusive access created by a **quiesce** command with either instance or database scope, it should not matter which timestamp is chosen for a PIT recovery.

If all other DB2 connections were forced off by the **quiesce database** or **quiesce instance** command, only one **quiesce tablespaces for table** command is needed in order to register a useful PIT in the recovery history file. This PIT is representative as a stable point of recovery for the whole database, and it does not matter which table the quiesce is taken against.

This concludes our testing of the **recover database** command with a single-partitioned database.

### ***Test 5: Recovery to PIT on partitioned database***

Our final testing is performed on a partitioned database. The test-bed that we use contains two logical partitions and two default partition groups (IBMCATGROUP, IBMDEFAULTGROUP) on a single DB2 instance.

The result of our test-bed preparation scripts cover the log and backup file ranges shown in Example 14-66.

#### *Example 14-66 First and last backup, log and table values in recovery range*

---

Backups are in: /db2/backup

First backup: TEST.0.db2inst1.NODE0000.CATN0000.20080501000855.001

Last backup: TEST.0.db2inst1.NODE0001.CATN0000.20080501021759.001

Archive logs (partition 0) are in: /db2/archlog/db2inst1/TEST/NODE0000/C0000000

First archive log: 2008-05-01 00:09 S0000000.LOG

Last archive log: 2008-05-01 02:18 S0000383.LOG

Archive logs (partition 1) are in: /db2/archlog/db2inst1/TEST/NODE0001/C0000000

First archive log: 2008-05-01 00:09 S0000000.LOG

Last archive log: 2008-05-01 02:18 S0000383.LOG

Active logs (partition 0) are in: /db2/data/db2inst0/NODE0000/SQL00001/SQLLOGDIR

First active log: 2008-05-01 02:15 S0000384.LOG

Last active log: 2008-05-01 02:18 S0000395.LOG

Active logs (partition 1) are in: /db2/data/db2inst0/NODE0001/SQL00001/SQLLOGDIR

First active log: 2008-05-01 02:14 S0000384.LOG

Last active log: 2008-05-01 02:18 S0000395.LOG

First timestamp in table at Testtab\_1\_256: 2008-05-01-00.18.06.475492

Last timestamp in table at Testtab\_127\_256: 2008-05-01-02.17.58.779885

---

This time we are aiming for the midpoint of the recovery range where the TESTTAB table column col2 value is Testtab\_63\_256. The output from our Script 2 execution tells us that the timestamp when this row was updated was 2008-05-01-01.16.19.296473 and the associated backup file has a timestamp of 20080501011620.

Before running our recover to PIT, we can see in Example 14-67 that DB2 has no trouble whatsoever recovering to end of logs with the default action of the recover database command, even on a partitioned database. The output of this command gives us a valuable timestamp which we can use later if we get into difficulties with recovery to PIT in multiple log chains.

*Example 14-67 Partitioned recovery - recover to end of logs*

---

```
db2inst1@lead:~> db2 recover db test
```

Rollforward Status

```
Input database alias           = test
Number of nodes have returned status = 2
```

Node number	Rollforward status	Next log to be read	Log files processed	Last committed transaction
0	not pending		S0000382.LOG-S0000383.LOG	2008-05-01-02.18.04.000000 Local
1	not pending		S0000382.LOG-S0000383.LOG	2008-05-01-02.18.01.000000 Local

---

```
DB20000I The RECOVER DATABASE command completed successfully.
```

---

To be thorough, for our primary test of recovering to a PIT shown in Example 14-68, we first attempt to recover to the point in time just after where we know the backup of both partitions occurred. It is by no means certain that we can find a consistent PIT that close to a backup file's timestamp for a partitioned database, and indeed, from the output we see that DB2 cannot find a satisfactory rollforward point at the timestamp we have specified. This is expected behavior, and we can take steps and follow heuristic methods to reach an acceptable PIT to recover to for both partitions.

*Example 14-68 Partitioned recovery - not a stable PIT*

---

```
db2inst1@lead:~> db2 "recover db test to 2008-05-01-01-16-21 on all dbpartitionnums"
SQL1262N The point-in-time specified for rolling forward database "test" is not valid.
```

---

Our next step covers some basic diagnosis. Running the same command again confirms that the recover command is failing in the middle of the rollforward operation. See Example 14-69.

*Example 14-69 Partitioned recovery - unstable PIT with more info*

---

```
db2inst1@lead:~> db2 "recover db test to 2008-05-01-01.16.21 on all dbpartitionnums"
SQL1275N The stoptime passed to roll-forward must be greater than or equal to "2008-05-01-01.16.24.000000 Local", because database "TEST" on node(s) "0" contains information later than the specified time.
```

---

In Example 14-70 we adjust the timestamp backwards to just before the backup timestamp, in an attempt to influence DB2 to use the previous set of database backup files and rollforward from there. This does not help us either, there is just no stable PIT in that time frame.

*Example 14-70 Partitioned recovery - another unstable PIT*

---

```
db2inst1@lead:~> db2 "recover db test to 2008-05-01-01.16.20 on all dbpartitionnums"
SQL1275N The stoptime passed to roll-forward must be greater than or equal to
"2008-05-01-01.16.24.000000 Local", because database "TEST" on node(s) "0,1"
contains information later than the specified time.
```

---

To be safe, we should retry with a slightly different PIT. DB2 might also need to be using an initial restore database using a backup image from a log chain chronologically prior to the one it is using now, which might be causing us difficulties with finding a stable PIT inside a certain timestamp range. In Example 14-71 we run another recover, effectively to the same log timestamp that we achieved with our initial default recover.

*Example 14-71 Partitioned recovery - restarting from the old log chain*

---

```
db2inst1@lead:~> db2 "recover db test to 2008-05-01-02.18.04 on all dbpartitionnums"
```

```

                                Rollforward Status

Input database alias              = test
Number of nodes have returned status = 2

Node number  Rollforward  Next log      Log files processed      Last committed transaction
              status      to be read
-----
              0  not pending
              1  not pending
              S0000190.LOG-S0000383.LOG  2008-05-01-02.18.04.000000 Local
              S0000190.LOG-S0000383.LOG  2008-05-01-02.18.01.000000 Local
```

```
DB20000I The RECOVER DATABASE command completed successfully.
```

---

At this stage, our database is effectively back to the state before we commenced that initial round of recover to PIT commands. In Example 14-72 we choose a timestamp with a half-second accuracy as close as possible after the col3 value we want was updated in the testtab table. We are hoping that this falls outside (before) the timestamp range of when partitioned database backup commences for that round of updates.

*Example 14-72 Partitioned recovery - a PIT before backups commence*

---

```
db2inst1@lead:~> db2 "recover db test to 2008-05-01-01.16.19.5 on all dbpartitionnums"
```

```

                                Rollforward Status

Input database alias              = test
Number of nodes have returned status = 2

Node number  Rollforward  Next log      Log files processed      Last committed transaction
              status      to be read
-----
              0  not pending
              1  not pending
              S0000190.LOG-S0000383.LOG  2008-05-01-01.16.19.500000 Local
              S0000190.LOG-S0000383.LOG  2008-05-01-01.16.19.500000 Local
```



```
0 not pending          S0000187.LOG-S0000189.LOG 2008-05-01-01.15.29.000000 Local
1 not pending          S0000187.LOG-S0000189.LOG 2008-05-01-01.16.19.000000 Local
```

DB20000I The RECOVER DATABASE command completed successfully.

---

So at least we are successful in recovery to PIT. But as to the result, we have to wait and see what values our table contains in Example 14-73:

#### *Example 14-73 Partitioned recovery - a successful result*

---

```
db2inst1@lead:~> db2 connect to test
```

```
Database Connection Information
```

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
db2inst1@lead:~> db2 "select * from testtab"
```

```
COL1      COL2      COL3
-----
1 Testtab_63_256 2008-05-01-01.16.19.296473
```

```
1 record(s) selected.
```

---

We see just what we wanted, the row value at col2's value of Testtab\_63\_256. We could not have planned that better if we tried. By using restore and rollforward, as well as manually determining timestamps of individual backup files and the PIT timestamp for the rollforward, this result would have taken us much longer to achieve.

That concludes our testing of the recover database command for both normal and partitioned databases.

## 14.4 Recovery object management

New to DB2 V9.5 is the ability to perform both automatic and simple command-driven pruning of recovery history data, archive log files, database backup files, and load copy files.

This new feature is highlighted in the DB2 9.5 Information Center in multiple locations:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.wn.doc/doc/c0051991.html>

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0051365.html>

Some functionality was introduced in DB2 9.1, such as the **update history** command, but enhanced in V9.5. Since DB2 V8, the **prune history** command has been available, also with less functionality than is available now. Backup to Tivoli Storage Manager (TSM) can still be managed with specification driven by the **db2adut1** command, but the recovery history file's DEVICE TYPE value of "A" for TSM now allows DB2 to manage those backup files natively in addition to other device types. Snapshot backup files created with DB2 Advanced Copy Services (ACS) should be managed with the **db2acsuti1** command.

Previously, sites have had to write custom scripts to achieve recovery object management, often re-inventing the wheel, discovering that a script on one platform did not work on another, or that migrating shell scripts from a UNIX crontab to a GUI-based DB2 Task Center is not necessarily as straightforward as it sounds.

For an overview of the concept of the recovery history file itself, we direct you to the Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0006136.htm>

For a general perspective of automatic recovery history file pruning, see:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0006138.htm>

## Feature summary

While most aspects of recovery object management are interrelated in DB2 9.5, we can split them conceptually into Automated and Command-driven categories.

### ***Automated recovery object management***

In "Test 2: PIT recovery" on page 771 from 14.3, "Recover database command" on page 759, we have seen how DB2 9.5 automatically removes older entries in the recovery history file. This basic management is controlled by two database configuration parameters, with DB2 automatically removing old entries based on the most restrictive of the two parameters:

- ▶ **NUM\_DB\_BACKUPS**: Expressed as number of backups  
Only this number of the most recent database backup file references in the recovery history file is kept.
- ▶ **REC\_HIS\_RETENTN**: Expressed as number of days  
Only this number of days worth of the most recent database backup file references in the recovery history file is kept.

For example, if NUM\_DB\_BACKUPS was set to 53, and REC\_HIS\_RETENTN was set to 366 based on an initial requirement of only keeping weekly database backup files for a year, then DB2 would either remove references to anything older than 366 days, or remove references to the older backups which brought the total number of backups higher than 53, whichever occurs first. This would not be an issue until some ad-hoc backups are performed in addition to the weekly backups, bringing the total in the recovery history file higher than 53 much sooner than the 366 days are up. Therefore, it is important to take these values into consideration for your site's specific backup retention requirements, in conjunction with behavioral changes in the dependency relationship of these two parameters explained in the paragraph for the next parameter.

Up to this point, we have only been referring to removing references to real objects, stored as entries in the recovery history file. The database configuration parameter, AUTO\_DEL\_REC\_OBJ, introduced in DB2 9.5 enables the automatic deletion of real recovery objects based on the values in the above two parameters. AUTO\_DEL\_REC\_OBJ takes two values:

- ▶ OFF: Only entries in the recovery history file are pruned to meet retention settings in NUM\_DB\_BACKUPS and REC\_HIS\_RETENTN.
- ▶ ON: DB2 also physically removes backup images, load copy images, and archive log files associated with those recovery history file entries. As a safety measure, when AUTO\_DEL\_REC\_OBJ is set to ON, the system only performs this maintenance when both the NUM\_DB\_BACKUPS *and* REC\_HIS\_RETENTN values are exceeded.

Given that the state of a database at certain milestones is desirable, and that certain backup images can be critical to long term recoverability, there is a way to explicitly protect individual recovery objects, dependant objects, and entries in the recovery history file by labelling them with a DO\_NOT\_DELETE flag. This topic is covered in the DB2 9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0051358.html>

As well as by the specific entry for the **update history file** command:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/r0001993.htm>

The **update history file** command actually has a tremendous potential in usability, from manually changing recovery object locations and device types in a disaster recovery scenario, to marking entries with a status of lost, corrupted, or unavailable (deleted), so recovery actions know to skip over those entries. Our interest in this section is the ability to mark a log or backup file entry with an X value, signifying a DB2HISTORY\_STATUS\_DO\_NOT\_DELETE entry status, preventing it from being removed by automated recovery history file pruning.

## **Command-driven recovery object management**

The primary means to prune old recovery objects in DB2 9.5 is the **prune history** command. This command has had a DB2 application programming interface (API) since at least DB2 Version 5, gradually expanded in scope and usefulness, without sacrificing a critical measure of safety for those objects. According to Information Center documentation going at least as far back as DB2 Version 6, the **prune history** command already supported automatic deletion of associated file backups on DB2 Data Links Manager servers. DB2 9.5's full support of pruning normal DB2 backups to disk among other devices is an extremely welcome addition that we have been waiting for. At DB2 Version 7, pruning physical log files required knowing the actual log file name, so being able to specify log file deletion by timestamp since DB2 V8 with the **prune history** command's **<timestamp>** and **delete** parameters made the command that much more usable and less error-prone.

Our reference for old DB2 APIs and data structures is in an application migration section of the DB2 Version 8 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/topic/com.ibm.db2.udb.doc/admin/r0009374.htm>

As an aside, it is an interesting point that DB2 Data Links Manager has been effectively providing end-users with relational locking and data integrity on what was essentially an abstract form of Binary Large Objects (BLOBs), well before that concept even made its way into DB2 as a native data format. It would appear that the age of this pioneering tool has come and gone, however, migration of DB2 from V8 to V9.5 requires removal of any existing Data Links Manager objects from existing databases:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.qb.migration.doc/doc/t0022646.html>

The default action of **prune history** remains the same in DB2 V9.5, to remove entries in the recovery history file, and nothing else. While dangerous enough in itself, it is still less risky than automatically removing associated physical log files. It is certainly much less risky than automatically removing associated backup files, which would have their own recovery history file and can also have their own included logs or can be offline backups, thus being completely self contained recovery objects from which a database can be restored. Some safety can be found in the **prune history** command's **with force option** parameter, which needs to be specified in order to remove anything in the active log chain more recent than the latest full database backup file was created. Needless to say, this action is rarely warranted, and places the database into a position where recovery to end of logs is no longer possible. It also removes entries and associated files even if the DB2HISTORY\_STATUS\_DO\_NOT\_DELETE flag has been set against the recovery history ENTRY\_STATUS column.

As we have mentioned, log files can be physically removed in addition to the recovery history entries by either using the **prune history** command's **<timestamp>** and **delete** parameters, or by the old **logfile prior to <log file name>** parameter.

Also, in DB2 9.5, using the **prune history** command's **<timestamp>** and **delete** parameters in conjunction with the db cfg parameter **AUTO\_DEL\_REC\_OBJ** set to **ON** allows you to delete actual backup files associated with obsolete recovery history file entries.

Given the numerous explicit command parameters and db cfg parameters required in order to perform removal of physical log files and backup files outside the recovery history file, the safety aspects of the **prune history** command are exemplary. While not completely foolproof, they result in users being in no doubt as to what effect running a given **prune history** command can have.

## Feature testing

Our plan for functional testing of these new features includes various combinations separated into categories of automated pruning, and pruning via explicit **prune history** commands.

### Preparation

We use the environment created in “Preparation for single-partitioned database recovery examples” on page 760 as the basis for all our testing here. This environment can be easily replicated by creating an empty database on any DB2 9.5 instance on an UNIX-based platform, configuring it for archive logging, then running the appropriate Script 1 and Script 2. Converting our shell scripts to batch or Java to run on a Windows DB2 platform is possible, we leave that for you as an optional exercise.

Our testing is carried out on a partitioned database for the sake of completeness. Though some commands need to specifically take this into account, all testing covered here is also applicable to non-partitioned environments.

What we expect to be seeing before running any of our tests is a database with a recovery history file containing many backup and log entries, in addition to a backup subdirectory and archive log subdirectory, each with many files.

Example 14-74 shows the range of our target recovery objects (archive log and database backup files).

#### *Example 14-74 The range of target backup and archive logs for pruning*

---

Backups are in: /db2/backup

First backup: TEST.0.db2inst1.NODE0000.CATN0000.20080501000855.001

Last backup: TEST.0.db2inst1.NODE0001.CATN0000.20080501021759.001

Archive logs (partition 0) are in: /db2/archlog/db2inst1/TEST/NODE0000/C0000000  
First archive log: 2008-05-01 00:09 S0000000.LOG  
Last archive log: 2008-05-01 02:18 S0000383.LOG

Archive logs (partition 1) are in: /db2/archlog/db2inst1/TEST/NODE0001/C0000000  
First archive log: 2008-05-01 00:09 S0000000.LOG  
Last archive log: 2008-05-01 02:18 S0000383.LOG

---

Rather than having to check individually which archive log files and backup files have been removed every time we perform recovery object pruning, we use a disk free command in Linux as a litmus test. At present, we are looking at disk space figures for the root file system on **/dev/sda2** shown in Example 14-75. On a real production DB2 system, these figures would be (hopefully) split out into separate specific file systems for archive logs, active logs, backup files, and the actual table space containers, so we would get more discrete indicators of free space.

*Example 14-75 How much disk space we have left at a glance*

---

```
db2inst1@lead:~> df -k
Filesystem            1K-blocks      Used Available Use% Mounted on
/dev/sda2              7807308    6901792    905516  89% /
udev                  385204         84    385120   1% /dev
```

---

Our range of testing is intended to be relatively short, but conceptually it is intended to show how multiple aspects of the new recovery object management features work, both using automatic, and manual methods with native DB2 commands to achieve these ends. Our first series of tests is entitled *Test 1*.

### **Test 1: Automatic pruning**

, we intend to show how DB2 automatically removes actual backup and log files in addition to entries in the recovery history file, while still protecting certain entries we do not want to be automatically deleted. This is accomplished by performing the following steps:

1. Set the retention period by updating the database configuration parameters.
2. Protect recovery history entries and associated backup and log files using the **update history** command.

First we take a look at our the current database configuration parameters for automatic recovery object management, shown in Example 14-76.

*Example 14-76 Our current settings for automatic recovery object management*

---

```
db2inst1@lead:~> db2 get db cfg for test|egrep "NUM_DB|_REC|HIS_"
Number of database backups to retain (NUM_DB_BACKUPS) = 366
```

Recovery history retention (days) (REC\_HIS\_RETENTN) = 366  
Auto deletion of recovery objects (AUTO\_DEL\_REC\_OBJ) = OFF

---

### ***Test 1: Protect recovery history file entries***

Before we start updating those database configuration parameters, our first priority is to explicitly protect a few key entries in our recovery history file. For demonstration purposes, we choose to protect the oldest backup file from our test-bed backup range, the backup file we used to recover to the mid-point in our previous recover to PIT testing, and a log file.

First we have to find the entry identifier (EID) for each of the recovery history file entries we wish to protect. The **update history** command only allows us to update status if we explicitly specify the EID as a parameter. We can get a list of EIDs easily enough through either the **list history** command, or using the administrative view SYSIBMADM.DB\_HISTORY.

To find what range of EIDs we are dealing with, the **min** and **max** functions shown in Example 14-77 are rather useful.

#### ***Example 14-77 Protect recovery history file entries - find EID range***

---

```
db2inst1@lead:~> db2 "select min(eid) as mineid,max(eid) as maxeid from sysibmadm.db_history"
```

```
MINEID          MAXEID
-----
1              728
```

```
1 record(s) selected.
```

---

Given that our EIDs range from 1 to 728 in order of oldest to youngest, and we want to protect the first database backup file entry from our test script, we select certain columns from the SYSIBMADM.DB\_HISTORY as shown in Example 14-78. We check the EIDs older than EID 10 to find our desired backup.

#### ***Example 14-78 Protect recovery history file entries - list EIDs older than 10***

---

```
db2inst1@lead:~> db2 "select char(char(eid),1) as e,char(char(dbpartitionnum),1) as p,operation as o,operationtype as t,start_time,char(rtrim(comment),23) as comment,char(rtrim(location),57) as location from sysibmadm.db_history where eid<10 order by 1,2"
```

```
E P O T START_TIME  COMMENT                                LOCATION
-----
1 0 B F 20080430233928 DB2 BACKUP TEST OFFLINE /db2/backup
1 1 B F 20080430233928 DB2 BACKUP TEST OFFLINE /db2/backup
2 0 X 1 20080501000904 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000000.LOG
2 1 X 1 20080501000901 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000000.LOG
3 0 B N 20080501000855 DB2 BACKUP TEST ONLINE /db2/backup
3 1 B N 20080501000855 DB2 BACKUP TEST ONLINE /db2/backup
4 0 X 1 20080501000905 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000001.LOG
```

```

4 1 X 1 20080501000905 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000001.LOG
5 0 X 1 20080501001239 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000002.LOG
5 1 X 1 20080501001241 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000002.LOG
6 0 X 1 20080501001806 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000003.LOG
6 1 X 1 20080501001719 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000003.LOG
7 0 X 1 20080501001812 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000004.LOG
7 1 X 1 20080501001811 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000004.LOG
8 0 B N 20080501001807 DB2 BACKUP TEST ONLINE /db2/backup
8 1 B N 20080501001807 DB2 BACKUP TEST ONLINE /db2/backup
9 0 X 1 20080501001813 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000005.LOG
9 1 X 1 20080501001815 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000005.LOG

```

---

18 record(s) selected.

That output shows us a brief list of our oldest backup and archive log file entries. Note the second column titled P, which indicates the partition number. If we had displayed an additional column named TBSPNAMES, we would see that our partitioning model has put the backup for SYSCATSPACE and SYSTOOLSPACE into partition 0, and has split USERSPACE1 into both partition 0 and partition 1. Another column we did not display here is DEVICETYPE, which for all our test-bed log and backup files, has a value of D for disk. From this output, it appears that the backup we are looking for is in EID 8. The first two backups (EID 1) are the offline backup that we ran to put the database into archive logging mode. The very first online backup (EID 3) is the one we ran in order to test that archive logging was working.

What we need to do at this point is update the value of a recovery history file column named ENTRY\_STATUS. For all our existing entries, the value is currently set to A (active). In order to protect an entry from deletion, the value should be set to X (do\_not\_delete). We do this now for EID 8 in Example 14-79. Note that the actual status value is case sensitive. The ENTRY\_STATUS column in the recovery history file can hold any of the following values: active, inactive, expired, deleted, or do\_not\_delete. The meaning of each of these values is explained in great detail here:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0051339.html>

*Example 14-79 Protect recovery history file entries - EID 8*

---

```

db2inst1@lead:~> db2 "update history eid 8 with status x"
SQL2032N The "piNewStatus" parameter is not valid. SQLSTATE=22531
db2inst1@lead:~> db2 "update history eid 8 with status X"
DB20000I The UPDATE HISTORY command completed successfully.

```

---

In a multi-partition database environment, the **update history** command only updates the recovery history file entry for the current partition, As shown in Example 14-80. only the ENTRY\_STATUS for partition 0 has been updated to X.



*Example 14-80 Protect recovery history file entries - check if update worked*

---

```
db2inst1@lead:~> db2 select eid,dbpartitionnum,entry_status from
sysibmadm.db_history where eid = 8
```

```
EID          DBPARTITIONNUM  ENTRY_STATUS
-----
              8             1 A
              8             0 X
```

2 record(s) selected.

---

To update all the database partitions, issue a **db2\_a11** command to run the update history specifically against all partitions, as shown in Example 14-81.

*Example 14-81 Protect recovery history file entries - db2\_all for partitions*

---

```
db2inst1@lead:~> db2_a11 "db2 connect to test;db2 update history eid 8 with
status X"
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
DB20000I The UPDATE HISTORY command completed successfully.
lead: db2 connect to test completed ok
```

Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
DB20000I The UPDATE HISTORY command completed successfully.
lead: db2 connect to test completed ok
```

```
db2inst1@lead:~> db2 "select entry_status from sysibmadm.db_history where eid=8"
```

```
ENTRY_STATUS
```

```
-----
X
X
```

2 record(s) selected.

---

## Protect the backup image files and log files

We also want to protect another backup file, and a log file. For simplicity's sake, we protect the log file at EID 6, as we have shown in Example 14-78. That log file should have no dependency relationship with the backup file we are protecting. As a result, we can see if DB2 protects log files that a given backup file depends on for rollforward recovery, or if we would have to explicitly update the EVENT\_STATUS value to X ourselves.

On our multi-partition environment, we protect the log file entries by updating the ENTRY\_STATUS column with the following command syntax:

```
db2_all "db2 connect to test;db2 update history eid 6 with status X"
```

The EID(s) for the backup file we want to protect can be found with a simple select with a predicate on the START\_TIME column, to match the timestamp in the database backup file name, shown in Example 14-82. Each database partition should have one backup file associated with it, unless file spanning has been specified in the **backup database** command. For file spanning, the EID is different for each file. There is an incremental SEQNUM and a unique LOCATION value to reflect the target for the backup files, but the START\_TIME is still the same for all those related backup files.

*Example 14-82 Protect recovery history file entries - finding midpoint backup EID.*

---

```
db2inst1@lead:~> select eid,operation as op from sysibmadm.db_history where
start_time='20080501011525'
```

```
EID          OP
----- --
                252 B
                254 B
```

2 record(s) selected.

---

We see here that while both those entries are for a backup (OPERATION 'B') which starts at the matching timestamp, there are different EID values. This attests to the complexity of managing recovery objects for multi-partitioned database, in that things which appear to be a single event on the surface, are in actuality multiple parallel chains with different events occurring at different times and taking different amounts of time to complete. We also discover in Example 14-83 that there are archive files directly associated with those two backup files, sharing the same EID.

*Example 14-83 Protect recovery history file entries - midpoint backup multiple EIDs.*

---

```
select char(char(eid),3),operation as op,char(rtrim(start_time),14) as
start_time,char(rtrim(end_time),14) as end_time from sysibmadm.db_history where
eid in (252,254)
```

```
1  OP START_TIME      END_TIME
--- -- -----
252 X 20080501011524 20080501011524
254 B 20080501011525 20080501011529
252 B 20080501011525 20080501011529
254 X 20080501011533 20080501011619
```

4 record(s) selected.

---

Having both logs and backups sharing an EID is not an issue. If other entries are associated with that backup, we definitely want to protect them anyway, as it indicates a dependency relationship.

Now that we know the EIDs, our partitioned database backup file is protected with these two commands:

```
db2_all "db2 connect to test;db2 update history eid 252 with status X"  
db2_all "db2 connect to test;db2 update history eid 254 with status X"
```

### ***Test 1: Update recovery object management db cfg parameters***

Our final step before triggering some automatic pruning of recovery objects is to adjust the three db cfg parameters that influence this DB2 9.5 feature.

When AUTO\_DEL\_REC\_OBJ is set to ON, and whenever there are more successful database backup entries than the NUM\_DB\_BACKUPS configuration parameter, then the database manager automatically prunes the recovery history file entries that are older than REC\_HIS\_RETENTN.

We update the AUTO\_DEL\_REC\_OBJ value to ON to enable automatic recovery object deletion of actual log and backup files. We update the NUM\_DB\_BACKUPS value to a more stringent value of 72 in order to keep an amount of backup files slightly older than our mid-point backup file, and to remove older unprotected entries.

At this stage, we do not update the REC\_HIS\_RETENTN value in order to test whether DB2 refuses to delete recovery history file entries unless they exceed both REC\_HIS\_RETENTN and NUM\_DB\_BACKUPS values while AUTO\_DEL\_REC\_OBJ is set to ON. We have already discovered that when AUTO\_DEL\_REC\_OBJ is set to OFF, DB2 deletes the entries based on whichever db cfg parameter is exceeded first. See 14.3, “Recover database command” on page 759 “Test 2: PIT recovery” on page 771. This concept is also described in the DB2 9.5 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006138.html>

Our configuration work is carried out in Example 14-84.

#### *Example 14-84 First round of update db cfg for automatic pruning*

---

```
db2inst1@lead:~> db2 update db cfg for test using AUTO_DEL_REC_OBJ ON  
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.  
db2inst1@lead:~> db2 update db cfg for test using NUM_DB_BACKUPS 72  
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
```

---

After a database deactivate, these values are confirmed as set in Example 14-85.

*Example 14-85 Checking db cfg parameters*

---

```
db2inst1@lead:~> db2 get db cfg for test | egrep "NUM_DB|_REC|HIS_"
Number of database backups to retain (NUM_DB_BACKUPS) = 72
Recovery history retention (days) (REC_HIS_RETENTN) = 366
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = ON
```

---

**Test 1: Automatic recovery object pruning**

Automatic pruning is triggered by a full database backup, or by a restore/rollforward database operation completing. As a simple demonstration, we just run another **backup database** command in Example 14-86.

*Example 14-86 Automatic recovery object pruning - trigger with backup.*

---

```
db2inst1@lead:/db2/backup> db2 backup db test on all dbpartitionnums to /db2/backup
compress
```

```
Part Result
```

```
-----
0000 DB20000I The BACKUP DATABASE command completed successfully.
0001 DB20000I The BACKUP DATABASE command completed successfully.
```

```
Backup successful. The timestamp for this backup image is : 20080502044500
```

---

Now we check whether anything is gone yet from the recovery history file, with the selection shown in Example 14-87.

*Example 14-87 Automatic recovery object pruning - checking recovery history.*

---

```
db2inst1@lead:/db2/backup> db2 "select min(eid) from sysibmadm.db_history where
operation='B'"
```

```
1
-----
1
```

---

As shown, nothing has gone yet; EID 1 is our very first backup entry. A quick look at our disk free space in Example 14-88 shows no increase in free space either, so no files have been deleted yet. It would appear that our backups and logs are not removed until we update the REC\_HIS\_RETENTN parameter to a more stringent value.

*Example 14-88 Automatic recovery object pruning - checking disk free space.*

---

```
db2inst1@lead:~> df -k
Filesystem          1K-blocks      Used Available Use% Mounted on
```

/dev/sda2	7807308	6901792	905516	89%	/
udev	385204	84	385120	1%	/dev

---

In Example 14-89, we update the final db cfg parameter, REC\_HIS\_RETENTN, to what might seem like a rash value of one day, however our previous test lab was only completed a day before over the course of less than an hour, so this is a necessary setting for our test environment.

*Example 14-89 Automatic recovery object pruning - update REC\_HIST\_RETENTN*

---

```
db2inst1@lead:~> db2 update db cfg for test using rec_his_retentn 1
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@lead:~> db2 get db cfg for test |egrep "NUM_DB|REC|HIS_"
Number of database backups to retain (NUM_DB_BACKUPS) = 72
Recovery history retention (days) (REC_HIS_RETENTN) = 1
Auto deletion of recovery objects (AUTO_DEL_REC_OBJ) = ON
```

---

To trigger the automatic pruning, we issue another backup database command in Example 14-90.

*Example 14-90 Automatic recovery object pruning - second trigger attempt.*

---

```
db2inst1@lead:/db2/backup> db2 backup db test on all dbpartitionnums to /db2/backup
compress
Part Result
---- -----
0000 DB20000I The BACKUP DATABASE command completed successfully.
0001 DB20000I The BACKUP DATABASE command completed successfully.
```

Backup successful. The timestamp for this backup image is : 20080502051205

---

Immediately, we see a significant result in Example 14-91. Our disk free space has changed dramatically.

*Example 14-91 Automatic recovery object pruning - check disk free space.*

---

```
db2inst1@lead:~> df -k
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/sda2        7807308    5688936   2118372   73% /
udev             385204         84     385120    1% /dev
```

---

We then check what has become of our precious protected entries for the backup and the log EID values of 8 and 6 respectively. We do this with another select statement in Example 14-92.

*Example 14-92 Automatic recovery object pruning - check protected entries.*

---

```
db2inst1@lead:/db2/backup> db2 connect to test
```

```
Database Connection Information
```

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
db2inst1@lead:/db2/backup> db2 "select min(eid) from sysibmadm.db_history where
operation='B'"
```

```
1
-----
                        8

1 record(s) selected.
```

```
db2inst1@lead:/db2/backup> db2 "select min(eid) from sysibmadm.db_history where
operation='X'"
```

```
1
-----
                        6

1 record(s) selected.
```

---

A whole swathe of backup files and archive log files have been physically removed off disk, without any explicit delete commands or scripting. Additionally, our protected entries remain unscathed.

Selecting a list of remaining database backup EIDs in our recovery history file shows that apart from our protected EID 8, only a limited number remain. See Example 14-93.

*Example 14-93 Automatic recovery object pruning - check all EIDs for backups*

---

```
db2inst1@lead:/db2/backup> db2 "select distinct eid from sysibmadm.db_history where operation='B'"
```

```
EID
-----
                        8
                        232
                        234
                        236
.
lots of values removed here for the benefit of your page-turning finger
.
                        727
                        729
                        730
147 record(s) selected.
```

---

While some might point out that 147 is not the same as the value of 72, which we specified in our NUM\_DB\_BACKUPS, we must hasten to add that one of those EIDs is protected outside the 72 limit, bringing the total to 146. Our other online partitioned backups have two EIDs per backup file, which brings the total number of backups down to 73, and the final pair of EIDs are for the offline database backups we only just performed to trigger the automatic recovery object management. They are both less than 24 hours old, so the value of 72 combined with the 1 day figure is correct, and completely affirms the description in the DB2 Information Center about how the recovery object management db cfg parameters behave and interact.

This concludes our testing of automatic recovery object management (*Test 1*). Our next series of tests is entitled *Test 2*.

### ***Test 2: Manual recovery object management***

Here we intend to show how obsolete backup and archive log files can be easily removed with the native DB2 prune history command, rather than requiring any special pattern-matching scripts external to DB2, and while safely avoiding the removal of protected recovery history file entries.

We build on and re-use our test environment as we left it at the end of “Test 1: Automatic pruning”. Our intent with this second round of tests is to further reduce our list of remaining backup and log files, beyond our protected mid-point, until we have only six of the most recent otherwise unprotected database backups and log files which are more than a day old.

In addition to EID 6 and 8, we expect to see our protected mid-point backup file left intact after this **prune history** command is issued.

Given that our test environment is ready to go, our goal here is to issue a **prune history** command to match the timestamp of the seventh most recent database backup. This enables us to keep the most recent six database backups. We choose the number six here arbitrarily, but it could easily represent six days of backups as a relevant analogy. Normally this value of six would be mapped to the REC\_HIS\_RETENTN db cfg parameter for automatic recovery object management.

For our partitioned database, each full database backup iteration consists of two physical files (one per partition). Keeping this in mind, we first request a list of the 20 most recent files in our backup subdirectory with a simple piped **ls** command in Example 14-94. This gives us a list of timestamps to choose from.

### Example 14-94 Prune history test - find a cut-off timestamp

---

```
db2inst1@lead:/db2/backup> ls -ltog | head -n 20
total 918891
-rw----- 1 15093760 2008-05-02 05:12 TEST.0.db2inst1.NODE0000.CATN0000.20080502051205.001
-rw----- 1 12079104 2008-05-02 05:12 TEST.0.db2inst1.NODE0001.CATN0000.20080502051205.001
-rw----- 1 15093760 2008-05-02 04:45 TEST.0.db2inst1.NODE0000.CATN0000.20080502044500.001
-rw----- 1 12079104 2008-05-02 04:45 TEST.0.db2inst1.NODE0001.CATN0000.20080502044500.001
-rw----- 1 12603392 2008-05-01 02:18 TEST.0.db2inst1.NODE0000.CATN0000.20080501021759.001
-rw----- 1 9850880 2008-05-01 02:18 TEST.0.db2inst1.NODE0001.CATN0000.20080501021759.001
-rw----- 1 12603392 2008-05-01 02:17 TEST.0.db2inst1.NODE0000.CATN0000.20080501021659.001
-rw----- 1 9850880 2008-05-01 02:17 TEST.0.db2inst1.NODE0001.CATN0000.20080501021659.001
-rw----- 1 12603392 2008-05-01 02:16 TEST.0.db2inst1.NODE0000.CATN0000.20080501021603.001
-rw----- 1 9850880 2008-05-01 02:16 TEST.0.db2inst1.NODE0001.CATN0000.20080501021603.001
-rw----- 1 12603392 2008-05-01 02:15 TEST.0.db2inst1.NODE0000.CATN0000.20080501021503.001
-rw----- 1 9850880 2008-05-01 02:15 TEST.0.db2inst1.NODE0001.CATN0000.20080501021503.001
-rw----- 1 12603392 2008-05-01 02:14 TEST.0.db2inst1.NODE0000.CATN0000.20080501021405.001
-rw----- 1 9850880 2008-05-01 02:14 TEST.0.db2inst1.NODE0001.CATN0000.20080501021405.001
-rw----- 1 12603392 2008-05-01 02:13 TEST.0.db2inst1.NODE0000.CATN0000.20080501021307.001
-rw----- 1 9850880 2008-05-01 02:13 TEST.0.db2inst1.NODE0001.CATN0000.20080501021307.001
-rw----- 1 12603392 2008-05-01 02:12 TEST.0.db2inst1.NODE0000.CATN0000.20080501021209.001
-rw----- 1 9850880 2008-05-01 02:12 TEST.0.db2inst1.NODE0001.CATN0000.20080501021209.001
-rw----- 1 12603392 2008-05-01 02:11 TEST.0.db2inst1.NODE0000.CATN0000.20080501021109.001
```

---

Because all entries with time stamps equal to or less than the time stamp provided are deleted from the recovery history file, if we want to keep the six most recent pairs of those database backup files, the timestamp we use as a cut-off point has to be less than that of the sixth most recent file pair.

We choose 20080501021405 as our timestamp, the same as the seventh most recent pair of backup files. This also tests whether DB2 does in fact remove all backups with timestamps less than or equal to our provided parameter.

Because we are using a partitioned database for our test-bed, we use the `db2_all` command wrapper to issue this **prune history** command, shown in Example 14-95.

### Example 14-95 prune history test - command for partitioned database

---

```
db2inst1@lead:/db2/backup> db2_all "db2 connect to test;db2 prune history 20080501021405 and delete"
```

#### Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
DB20000I The PRUNE command completed successfully.
lead: db2 connect to test completed ok
```

#### Database Connection Information

```
Database server      = DB2/LINUX8664 9.5.0
SQL authorization ID = DB2INST1
Local database alias = TEST
```

```
DB20000I The PRUNE command completed successfully.
lead: db2 connect to test completed ok
```

---



Looking at our /db2/backup subdirectory contents in Example 14-96, we do not even need to truncate the output, there are only our six recent database backups, plus the two earlier protected database backups.

*Example 14-96 prune history test - listing backup files remaining*

---

```
db2inst1@lead:/db2/backup> ls -ltog
total 194446
-rw----- 1 15093760 2008-05-02 05:12 TEST.0.db2inst1.NODE0000.CATN0000.20080502051205.001
-rw----- 1 12079104 2008-05-02 05:12 TEST.0.db2inst1.NODE0001.CATN0000.20080502051205.001
-rw----- 1 15093760 2008-05-02 04:45 TEST.0.db2inst1.NODE0000.CATN0000.20080502044500.001
-rw----- 1 12079104 2008-05-02 04:45 TEST.0.db2inst1.NODE0001.CATN0000.20080502044500.001
-rw----- 1 12603392 2008-05-01 02:18 TEST.0.db2inst1.NODE0000.CATN0000.20080501021759.001
-rw----- 1 9850880 2008-05-01 02:18 TEST.0.db2inst1.NODE0001.CATN0000.20080501021759.001
-rw----- 1 12603392 2008-05-01 02:17 TEST.0.db2inst1.NODE0000.CATN0000.20080501021659.001
-rw----- 1 9850880 2008-05-01 02:17 TEST.0.db2inst1.NODE0001.CATN0000.20080501021659.001
-rw----- 1 12603392 2008-05-01 02:16 TEST.0.db2inst1.NODE0000.CATN0000.20080501021603.001
-rw----- 1 9850880 2008-05-01 02:16 TEST.0.db2inst1.NODE0001.CATN0000.20080501021603.001
-rw----- 1 12603392 2008-05-01 02:15 TEST.0.db2inst1.NODE0000.CATN0000.20080501021503.001
-rw----- 1 9850880 2008-05-01 02:15 TEST.0.db2inst1.NODE0001.CATN0000.20080501021503.001
-rw----- 1 12603392 2008-05-01 01:15 TEST.0.db2inst1.NODE0000.CATN0000.20080501011525.001
-rw----- 1 9850880 2008-05-01 01:15 TEST.0.db2inst1.NODE0001.CATN0000.20080501011525.001
-rw----- 1 12603392 2008-05-01 00:18 TEST.0.db2inst1.NODE0000.CATN0000.20080501001807.001
-rw----- 1 9850880 2008-05-01 00:18 TEST.0.db2inst1.NODE0001.CATN0000.20080501001807.001
```

---

The DB2 archive log files in our file system have also been drastically reduced. For each database partition (subdirectories NODE0000 and NODE0001), the only remaining log files we have are:

- ▶ S0000003.LOG from the first protected EID 6.
- ▶ S0000186.LOG and S0000189.LOG as dependant log objects from our second protected backups on EID 252 and 254.
- ▶ Active log files subsequent to S0000370.LOG in the current log chain. Our test platform had accumulated five log chains due to previous recoveries performed in our test environment.

An extract of the recovery history file contents in Example 14-97 shows what is remaining there. Some entries are related to DB2 performing automatic crash recovery after we resumed our suspended virtual image on our test-bed platform. Our primary interest are EIDs 6, 8, 252, and 254, that would all have been deleted if we had not protected them. They would have been deleted despite being protected if we had specified **prune history <timestamp> with force option**. All subsequent entries are from a timestamp more recent than our limit of 20080501021405 which we specified in our **prune history** command.

*Example 14-97 prune history test - list EIDs remaining*

---

```
db2inst1@lead:/> db2 "select substr(digits(eid),17,3) as eid,char(char(dbpartitionnum),1) as
p,operation as o,operationtype as t,start_time,char(rtrim(comment),23) as
comment,char(rtrim(location),57) as location from sysibmadm.db_history order by 1,2"
EID P O T START_TIME COMMENT LOCATION
-----
006 0 X 1 20080501001806 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000003.LOG
006 1 X 1 20080501001719 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000003.LOG
```

```

008 0 B N 20080501001807 DB2 BACKUP TEST ONLINE /db2/backup
008 1 B N 20080501001807 DB2 BACKUP TEST ONLINE /db2/backup
252 0 X 1 20080501011524 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000186.LOG
252 1 B N 20080501011525 DB2 BACKUP TEST ONLINE /db2/backup
254 0 B N 20080501011525 DB2 BACKUP TEST ONLINE /db2/backup
254 1 X 1 20080501011533 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000189.LOG
495 1 X 1 20080501021406 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000370.LOG
497 0 X 1 20080501021409 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000370.LOG
497 1 X 1 20080501021411 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000371.LOG
498 1 X 1 20080501021412 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000372.LOG
499 0 X 1 20080501021411 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000371.LOG
499 1 X 1 20080501021506 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000373.LOG
500 0 X 1 20080501021502 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000372.LOG
500 1 B N 20080501021503 DB2 BACKUP TEST ONLINE /db2/backup
501 0 X 1 20080501021507 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000373.LOG
501 1 X 1 20080501021509 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000374.LOG
502 0 B N 20080501021503 DB2 BACKUP TEST ONLINE /db2/backup
502 1 X 1 20080501021511 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000375.LOG
503 0 X 1 20080501021509 - /db2/archlog/db2inst1/TEST/NODE0000/C0000000/S0000374.LOG
503 1 X 1 20080501021605 - /db2/archlog/db2inst1/TEST/NODE0001/C0000000/S0000376.LOG
---
---irrelevant lines removed here - related to DB2 crash recovery on our test-bed platform
---
726 1 B F 20080502044500 DB2 BACKUP TEST OFFLINE /db2/backup
727 0 F P 20080501140550 20080501210553GMT -
727 1 B F 20080502051205 DB2 BACKUP TEST OFFLINE /db2/backup
728 0 X 1 20080501140554 - /db2/archlog/db2inst1/TEST/NODE0000/C0000004/S0000383.LOG
729 0 B F 20080502044500 DB2 BACKUP TEST OFFLINE /db2/backup
730 0 B F 20080502051205 DB2 BACKUP TEST OFFLINE /db2/backup

468 record(s) selected.

```

Final vindication for our having run the prune history command and automatic recovery object management is in our current free disk space value, shown in Example 14-98.

*Example 14-98 prune history test - disk free space*

```

db2inst1@lead: /> df -k
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/sda2        7807308    4229468   3577840   55% /
udev             385204         84     385120    1% /dev

```

Remembering that we were using compressed backups, and our database was essentially empty with very little logging occurring, our overall space savings as a direct result of these two tests amount to 3577840 - 905516 => 2672324 kilobytes, or about 2.6 gigabytes. That is quite substantial, and shows how automatic recovery object management in DB2 9.5 is a crucial time-saver for sites who cannot afford to waste time on housekeeping for their own proto-typing or non-production environments, and for those who continually run into “file system full” types of errors resulting in otherwise unnecessary calls to technical support staff.



# HACMP application server scripts

In this appendix we provide application server scripts used in an HADR with HACMP environment. The application server start and stop scripts are used to automate the HADR takeover.

## A.1 hadr\_primary\_takeover.ksh

Example A-1 shows a sample script for HADR takeover in an HADR with HACMP environment.

*Example: A-1 hadr\_primary\_takeover.ksh*

---

```
#!/usr/bin/ksh -x

#####
# hadr_primary_takeover.ksh
# This script is called when Rotating Resource Group starts.
# Skip processes in NORMAL HACMP START UP.
# Issue hadr takeover when fail over case.
#####

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

set -x

#####
# set PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# set START_MODE
#####
# LOCAL=HOMELocal!=HOME
# PRE=""NOMALFALLOVER(manual online operation)
# PRE!=""FALLBACKFALLOVER

### /usr/es/sbin/cluster/utilities/cLRGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo HACMP V5.2 must be running
### exit 1
### fi

/usr/es/sbin/cluster/utilities/cLRGinfo -a

/usr/es/sbin/cluster/utilities/cLRGinfo -a | grep $R_RG \
| awk -F\ " '{print $2}' | awk -F\: '{print $1}' | read PRENODE
```

```

if [[ "$PRENODE" = "" ]]; then
    START_MODE="NORMAL"
else
    START_MODE="FALLOVER"
fi

#####
# MONITOR HADR ROLE & STATE
#####
#Get HADR State,HADR Role(snapshot/db cfg)
${HADR_MONITOR} ${DB2HADRINSTANCE1} ${DB2HADRINSTANCE2} ${DB2HADRDBNAME}
${VERBOSE}| read ROLE_CFG_LOCAL STATE_SNP_LOCAL ROLE_SNP_LOCAL

#####
# MAIN
#####
case ${START_MODE} in
NORMAL)
    echo "Start trigger is Normal startup: Skip HADR operations"
;;
FALLOVER)
    echo "Start trigger is Takeover: Check HADR role & state"

    if [[ "${ROLE_SNP_LOCAL}" = "Primary" ]]; then
        # already primary
        echo "HADR role is already Primary:Skip HADR operations. When you stop
HACMP in takeover mode, stop script changes HADR role before this script is
issued."
    # elif [[ "${ROLE_SNP_LOCAL}" = "Standby" ]]; then
    #     # this instance is standby
    #     # try takeover by force
    #     echo "HADR role is Standby: Execute HADR TAKEOVER BY FORCE"
    #
    #     ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME} by force"

    elif [[ "${ROLE_SNP_LOCAL}" = "Standby" && "${STATE_SNP_LOCAL}" = "Peer"
]]; then
        # this instance is standby peer
        # try takeover by force
        echo "HADR role is Standby and HADR status is Peer: Execute HADR
TAKEOVER BY FORCE:"
        ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME} by force"

    elif [[ "${ROLE_SNP_LOCAL}" = "Standby" && "${STATE_SNP_LOCAL}" != "Peer"
]]; then

```

```

        # this instance is not standby peer
        echo "HADR role is Standby but HADR status is Not Peer: You might lose
data by takeover for inconsistency of databases, Skip HADR operations."

    else
        echo "HADR role is neithe Primary or Standby: Skip HADR operations "

    fi
;;

esac

date

exit 0

```

---

## A.2 hadr\_primary\_stop.ksh

Example A-2 shows a sample stop script for HADR takeover in an HADR with HACMP environment.

*Example: A-2 hadr\_primary\_stop.ksh*

---

```

# Normal stop operation: No operation is done.
# Stop in takeover mode: If HADR Primary & Peer, execute hadr takeover on
Standby by remote command
#####

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

set -x

#####
# set PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# set STOP_MODE
#####

```

```

# LOCAL=HOMELOCAL!=HOME
# PRE=""NOMALFALLOVER(manual online operation)
# PRE!="FALLBACKFALLOVER

### /usr/es/sbin/cluster/utilities/c1RGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo HACMP V5.2 must be running
### exit 1
### fi

/usr/es/sbin/cluster/utilities/c1RGinfo -a

/usr/es/sbin/cluster/utilities/c1RGinfo -a | grep $R_RG \
| awk -F\" '{print $2}' | awk -F\": '{print $2}' | read POSTNODE

if [[ "$POSTNODE" = "" ]]; then
    STOP_MODE="NORMAL"
else
    STOP_MODE="FALLOVER"
fi

#####
# MONITOR HADR ROLE & STATE
#####
#Get HADR State, HADR Role (snapshot, db cfg)
${HADR_MONITOR} ${DB2HADRINSTANCE1} ${DB2HADRINSTANCE2} ${DB2HADRDBNAME}
${VERBOSE} | read ROLE_CFG_LOCAL STATE_SNP_LOCAL ROLE_SNP_LOCAL

#####
# MAIN
#####
case ${STOP_MODE} in
NORMAL)
    echo "Stop HACMP in graceful mode"
;;
FALLOVER)
    echo "Stop HACMP in takeover mode "

    if [[ "${ROLE_SNP_LOCAL}" = "Primary" && "${STATE_SNP_LOCAL}" = "Peer" ]];
then
        echo "HADR takeoveris issued on remote node"
        /usr/es/sbin/cluster/sbin/c1_nodecmd -cspoc "-n ${REMOTENODE}" ${SU_CMD}
        "db2 takeover hadr on db ${DB2HADRDBNAME}"
    fi
;;
esac

```

```

date

exit 0

env.sh

DB2HADRINSTANCE1=hadrinst      # replace db2instp with P's instance name
DB2HADRINSTANCE2=hadrinst      # replace db2insts with S's instance name
DB2HADRDBNAME=sample           # replace hadrdb with HADR db name
VERBOSE=verbose                 # change to verbose to get more diagnostics
logged
R_RG=hadr_rg                    rotating resource group name
C_RG=db2_rg                     concurrent resource group name

SU_CMD="su - ${DB2HADRINSTANCE1} -c"      #su command
LOCALNODE=~ /usr/es/sbin/cluster/utilities/get_local_nodename`      #LOCAL
NODE
#REMOTENODE=~ /usr/es/sbin/cluster/utilities/clrsctinfo -cp cllsif | grep -v
${LOCALNODE} | cut -f 6 -d ":" | uniq`      #REMOTE NODE

LANG=C

/usr/bin/env LANG=C /usr/es/sbin/cluster/utilities/clshowres -g "${R_RG}" |
grep "Participating Node Name(s)" | awk '{print $4 " " $5}' | read n1 n2

if [[ "${n1}" = "${LOCALNODE}" ]];then
    REMOTENODE=${n2}
else
    REMOTENODE=${n1}
fi

HADR_MONITOR="/home/hadrinst/scripts/hadr_monitor.ksh"      # Get HADR State
HADR Role (snapshot, db cfg)
#! /usr/bin/ksh

```

---



## A.3 hadr\_monitor.ksh

Example A-3 shows a sample script for monitoring the HADR state.

*Example: A-3 hadr\_monitor.ksh*

---

```
# hadr_monitor.ksh
#####
# hadr_monitor.ksh
# This script is called by Concurrent Resource Group start script,
# Rotate Resource Group starts/stops script,
# and application monitor.
#
# 1.Get HADR role and state by database snapshot
# 2.Get HADR role by database configurations
#####
set -x

#####
# SET PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# GET HADR ROLE & STATE
#####

HADR_PROBE=~${SU_CMD} "db2 get snapshot for database on ${DB2HADRDBNAME}" | awk
'($1 == "State" && $2 == "=") || ($1 == "Role" && $2 == "=")' | sort | awk
'{print $3}'`
echo $HADR_PROBE | read hadr_role hadr_state other

if [[ "${hadr_state}" = "" || "${other}" != "" ]];then
    hadr_role=$( ${SU_CMD} "db2 get snapshot for database on ${DB2HADRDBNAME}"
| awk ' $1 == "Role" && $2 == "=" {print $3}' )
    hadr_state=$( ${SU_CMD} "db2 get snapshot for database on ${DB2HADRDBNAME}"
| awk ' $1 == "State" && $2 == "=" {print $3}' )
fi

hadr_role_cfg=$( ${SU_CMD} "db2 get db cfg for ${DB2HADRDBNAME} | grep 'HADR
database role' | awk '{print \$5}'")

echo $hadr_role_cfg $hadr_state $hadr_role

exit 0
```

---



## **DB2 and TSA configuration for ssh**

In this appendix we show how to configure DB2 and TSA to use ssh.

## B.1 Using DB2 with SSH

This URL explains in great detail how to go about setting up DB2 to use SSH authentication rather than the old default RSH, as used by commands such as `db2gcf`.

<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0506finnie/index.html>

The following link in the DB2 V9.1 Information Center sets out the basic requirements for changing DB2 remote commands from using RSH to SSH.

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.admin.doc/doc/c0005828.htm>

Essentially, after having set up SSH password-less remote commands, the DB2 registry variable `DB2RSHCMD` should be set to the path where the `ssh` executable binary is located, for example:

```
db2set -i db2inst1 DB2RSHCMD=/usr/bin/ssh
```

After recycling that DB2 instance, this registry variable should take effect for all remote commands issued against that instance.

## B.2 Using TSA with SSH

When installed into the default location, DB2 V9.1 contains TSA-specific registration and management scripts in:

```
/opt/ibm/db2/V9.1/ha/tsa
```

DB2 V8.2 has these scripts in the default location of:

```
/opt/IBM/db2/V8.2/ha/salinux
```

Because this location is not generally set up with a path, use of certain initialization scripts such as `regdb2sa1in` necessitates use of explicit path names.

Example B-1 shows the diagnostic output from a case-insensitive search on “rsh” in the TSA High Availability scripts subdirectory within the DB2 product libraries.

*Example: B-1 Output showing which TSA-DB2 scripts refer to “rsh”*

```
db2inst0@lead:/opt/ibm/db2/V9.1/ha/tsa> for a in *;do echo $a;cat $a |grep -i
rsh;done
das_monitor.ksh
das_start.ksh
das_stop.ksh
db2_monitor.ksh
db2_start.ksh
db2_stop.ksh
getstatus
hadr_monitor.ksh
hadr_start.ksh
hadr_stop.ksh
mount_monitor.ksh
mount_start.ksh
mount_stop.ksh
regdb2salin
SSH_OR_RSH=rsh
    printf " -s,[optional] use ssh (default rsh)\n"
        # Use ssh instead of default rsh
        SSH_OR_RSH=ssh
        echo "Distributing policy to node $CLUSTER_HOST via $SSH_OR_RSH..."
        ssh_or_rsh_is_ok=$(SSH_OR_RSH $CLUSTER_HOST "echo 0")
        if [[ -z $ssh_or_rsh_is_ok ]]; then
            echo "Error: cannot $SSH_OR_RSH to $CLUSTER_HOST"
            SSH_OR_RSH $CLUSTER_HOST "/bin/mkdir -p ${inst_path_salin?};/bin/chmod
755 ${inst_path_salin?}"
            SSH_OR_RSH $CLUSTER_HOST "/bin/chmod a+x ${inst_path_salin?}/*"
            SSH_OR_RSH $CLUSTER_HOST "/bin/ls ${mount_start} ${mount_stop}
${mount_monitor} ${db2_start} ${db2_stop} ${db2_monitor} ${das_start}
${das_stop} ${das_monitor} | wc -l" > /tmp/a 2> /tmp/b
            rc=$(SSH_OR_RSH $CLUSTER_HOST "${mount_stop?} ${mpthis?} ")
            ssh_or_rsh_is_ok=$(SSH_OR_RSH $CLUSTER_HOST "echo 0")
            if [[ -z $ssh_or_rsh_is_ok ]]; then
                echo "Error: cannot $SSH_OR_RSH to $CLUSTER_HOST"
                SSH_OR_RSH $CLUSTER_HOST "${db2_path_bin?}/db2greg -delvarrec
variable=hostname" 2> /dev/null > /dev/null
                SSH_OR_RSH $CLUSTER_HOST "${db2_path_bin?}/db2greg -delvarrec
variable=switchname" 2> /dev/null > /dev/null
                SSH_OR_RSH $CLUSTER_HOST "${db2_path_bin?}/db2greg -addvarrec
variable=hostname,value=${CLUSTER_HOST}"
                SSH_OR_RSH $CLUSTER_HOST "${db2_path_bin?}/db2greg -addvarrec
variable=switchname,value=${CLUSTER_HOST}"
                SSH_OR_RSH $CLUSTER_HOST "/bin/mkdir -p $mpthis" > /tmp/a 2>
/tmp/b
```

```

                $SSH_OR_RSH $CLUSTER_HOST "/bin/chmod 755 $mpthis" > /tmp/a
2> /tmp/b
                remoteNodediskID=$(($SSH_OR_RSH $CLUSTER_HOST "lspv | grep
$diskID | cut -c 1-10 ")
                $SSH_OR_RSH $CLUSTER_HOST "${mount_stop?} ${mpthis?}"
                $SSH_OR_RSH $CLUSTER_HOST "/usr/sbin/varyoffvg $vgname 2>
/dev/null > /dev/null"
                $SSH_OR_RSH $CLUSTER_HOST "/usr/sbin/chvg -a n $vgname 2>
/dev/null > /dev/null"
                $SSH_OR_RSH $CLUSTER_HOST "/usr/sbin/exportvg $vgname 2>
/dev/null > /dev/null"
                $SSH_OR_RSH $CLUSTER_HOST "/usr/sbin/importvg -y $vgname
-V${majNum?} ${remoteNodediskID?}" 2> /tmp/$vgname > /dev/null
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${mount_start?} ${mpthis?} ")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${mount_monitor?} ${mpthis?}
verbose") 2> /dev/null
                $SSH_OR_RSH $CLUSTER_HOST "su - ${INSTNAME} -c 'db2set
DB2_NUM_FAILOVER_NODES=' " > /dev/null
                $SSH_OR_RSH $CLUSTER_HOST "su - ${INSTNAME} -c 'db2set
DB2_NUM_FAILOVER_NODES=${no_nodes_cfg_line_no?}' > /dev/null"
                $SSH_OR_RSH $CLUSTER_HOST "su - ${INSTNAME} -c 'db2 update dbm cfg using
START_STOP_TIME 2' > /dev/null" > /dev/null
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${db2_start?} ${DB2INSTANCE?} $tnn")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${db2_monitor?} ${DB2INSTANCE?} $tnn
verbose")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${db2_stop?} ${DB2INSTANCE?} $tnn")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${das_start?} ${DB2INSTANCE?}")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${das_monitor?} ${DB2INSTANCE?}
verbose")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${das_stop?} ${DB2INSTANCE?}")
                rc=$(($SSH_OR_RSH $CLUSTER_HOST "${mount_stop?} ${mpthis?}")
ssh_or_rsh_is_ok=$(($SSH_OR_RSH $NODENAME "echo 0")
if [[ -z $ssh_or_rsh_is_ok ]]; then
    echo "Error: cannot $SSH_OR_RSH to $NODENAME"
    $SSH_OR_RSH $NODENAME "${db2_path_bin?}/db2greg -delvarrec
variable=hostname" 2> /dev/null > /dev/null
    $SSH_OR_RSH $NODENAME "${db2_path_bin?}/db2greg -addvarrec
variable=hostname,value=${NODENAME}"
    $SSH_OR_RSH $NODENAME "${db2_path_bin?}/db2greg -delvarrec
variable=switchname" 2> /dev/null > /dev/null
    $SSH_OR_RSH $NODENAME "${db2_path_bin?}/db2greg -addvarrec
variable=switchname,value=${NETNAME}"
reghadrsalin
unregdb2salin
unreghadrsalin

```

---

Only the regdb2salin script contains references to rsh.

On further inspection, there is a global variables section near the header comments of the regdb2salin file, containing the text in Example B-2.

*Example: B-2 Global variable section of regdb2salin*

---

```
# Globals...
PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/sbin/rsct/bin
export PATH
VERBOSE=false
SSH_OR_RSH=rsh
SCP_OR_RCP=rcp
UNAME=$(uname)
MYID=root
```

---

By default, the unsecure rsh is set as the remote shell method, and rcp as default remote copy. These can be changed by having the root user ID editing the initial setting in regdb2salin on both nodes from:

```
SSH_OR_RSH=rsh
SCP_OR_RCP=rcp
```

to:

```
SSH_OR_RSH=ssh
SCP_OR_RCP=scp
```

The SSH\_OR\_RSH parameter can also be changed at runtime with use of the -s flag to force usage of ssh.

For diagnostic purposes, the VERBOSE=false can be changed to VERBOSE=true, or better still, just use the -v flag as a **regdb2salin** command parameter.

Once registered within TSA, the other scripts are controlled by and executed from the TSA software, so the initial setting to use ssh or rsh is carried over. For security and integrity purposes, all the DB2 TSA scripts are owned by root:root, with permissions of 555 (r-xr-xr-x), on the files, and on the parent subdirectory structure, which can make backing up and even editing these files problematic.

Executing regdb2salin in verbose mode sheds light on what commands are being issued if a problem occurs. For example, at the first time an attempt is made to start DB2, this command is executed:

```
ssh lead '/usr/sbin/rsct/sapolicies/db2/db2_start.ksh db2inst0 0'
```

If this step hangs or fails, messages will appear in db2diag.log. However, nothing may happen in the regdb2salin script.

Inside that db2\_start.ksh script, this command has been executed:

```
/home/db2inst0/sqllib/bin/db2gcf -t 60 -u -p 0 -i db2inst0
```

Extrapolating those parameters in the command reference section of the DB2 Information Center, we can see that a request is being made to bring partition 0 of the db2inst0 instance up, with a timeout value override of 60 seconds. (Partition numbers are ignored for the db2gcf command in a non-partitioned environment).

Due to issues that might otherwise arise later when first executing regdb2salin, it is a good idea to have the root user ID test starting and stopping the DB2 instance with the db2gcf command prior to running regdb2salin, as shown in Example B-3.

*Example: B-3 Testing db2gcf command to confirm functionality using ssh*

---

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -s -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 State : Operable
```

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -u -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 Start : Success
```

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -s -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 State : Available
```

```
lead:~ # /home/db2inst0/sqllib/bin/db2gcf -d -i db2inst0
```

```
Instance : db2inst0
```

```
DB2 Stop : Success
```

---

One other example of regdb2salin failing that we discovered with this scenario of using two different DB2 instance user IDs, was that regdb2salin still expected and required both those DB2 instance user IDs to be on both nodes, even if only one of each had an associated DB2 instance created on it. This in turn necessitated sharing all public ssh keys in a four-way combination between those two IDs on both nodes. Once this was established, regdb2salin worked flawlessly, but executing regdb2salin in both verbose and non-verbose mode was invaluable in establishing the cause of the error.



A partial or failed execution of `regdb2salin` can be restarted after cleaning up definitions with the `unregdb2salin` command, as shown in Example B-4. Note this does not remove the network interface equivalencies that were set up in a previous step.

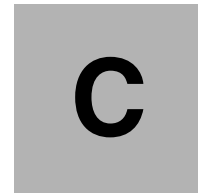
*Example: B-4 Cleaning up a failed `regdb2salin` execution*

---

```
Lead:/opt/ibm/db2/V9.1/ha/tsa # ./unregdb2salin -a db2inst0
Stopping all online resource groups for instance db2inst0...
Removing all relationship's ...
Removing equivalency's ...
Removing all resources ...
Removing all resource groups ...
Lead:/opt/ibm/db2/V9.1/ha/tsa # lsequ
Displaying Equivalencies :
virpubnic_lochnese
virpubnic_lead
```

---





## TSA scripts

In this appendix we provide custom takeover scripts used in the HADR with TSA environment of 11.3, “Automating HADR takeover with TSA on AIX” on page 377.

The start, monitor, and stop scripts are used to automate the HADR takeover.

## C.1 env file

Example C-1 shows an env definition file in an HADR with TSA environment.

*Example: C-1 env file*

---

```
#####
#
# FOR PRIMARY NODE
#
#####
LANG=C

# HADR DEFINITION
DB2HADRINSTANCE1=db2inst1# HADR Primary instance name
DB2HADRINSTANCE2=db2inst1# HADR Standby instance name
DB2HADRDBNAME=sample# HADR database name
HADR_RG=hadr_rg# Resource group name starts on both node

#SU DEFINITION
SU_CMD="su - ${DB2HADRINSTANCE1} -c"# su command to instance owner
SU_CMD2="su - ${DB2HADRINSTANCE2} -c"# su command to instance owner
DATE="`date +%Y-%m-%d-%H.%M.%S`"# date command

# GET THE NODE NAME OF BOTH NODE
LOCALNODE=`Baltic`
# Specify the hostname of the remote node.
REMOTENODE="Zaire"
#REMOTENODE=`lsrpnode | awk '{print $1}' | grep -v $LOCALNODE | grep -v Name`

# REMOTE SHELL DEFINITION
RSH="ssh ${REMOTENODE}"# RSH command using SERVICE N/W
RSH_H="ssh ${REMOTENODE}_h"# RSH command using HADR N/W

# PROGRAM DEFINITION
PROGDIR=`dirname $0`

#WRITE_STATUS="$PROGDIR/write_status.sh"#The scripts that writes the HADR status to the
standby node by rsh.
#TAIL_HADR_STATUS="$PROGDIR/tail_hadr_status.sh"#The scripts that tails db2diag.log and
monitors the HADR status.

# FLAG DEFINITION
FLAGDIR="/tmp"
#STATUS_FLAG="$FLAGDIR/status.flag"#The flag file of the HADR status
HADR_RG_START_FLAG="$FLAGDIR/hadr_rg_start.flag"#The flag file created when HADR_RG
starts
PLANNED_TKO_FLAG="$FLAGDIR/planned_tko.flag"#This flag should be created by user before
planned takeover

# MODIFY DIAGLOG PATH -- You can get it by db2 get dbm cfg | grep "DIAGPATH"
DIAGLOG="/home/${DB2HADRINSTANCE1}/sql1lib/db2dump/db2diag.log"
```

---

## C.2 hadr\_start.ksh

Example C-2 shows a sample script for HADR takeover in an HADR with TSA environment.

*Example: C-2 hadr\_start.ksh file*

---

```
#!/bin/ksh -x

#####
# hadr_start.sh
# This script is called when HADR_RG starts.
#
# #1. Start the script tails db2diag.log.
# 2. Execute HADR TAKEOVER if the role is STANDBY judging the status.
#####

PROGNAME=`basename $0`

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

#####
# SET PARAMETER
#####
. `dirname $0`/env
. `dirname $0`/get_hadr_info.fnc

#####
# GET HADR STATE, HADR ROLE(SNAPSHOT and DB CFG) ON LOCAL NODE
#####
F_get_status_by_db2pd

## Check PRIMARY_HADR_STATUS Flag file
PRIMARY_HADR_STATUS=$( tail -n 1 ${STATUS_FLAG} | awk '{print $2}' )

#####
# MAIN
#####

RC=0

#####
## START THE SCRIPT THAT TAILS DB2DIAG.LOG
#####
## Start the script if the tail process is not run.
ps -ef | grep "tail -n0" | grep -v "grep"
if [[ $? != 0 ]]; then
```

```

# echo "Start the script that tails db2diag.log"
# logger -i -p info -t $PROGNAME "Start the script that tails db2diag.log"
#
# ${TAIL_HADR_STATUS} 2>&1 /dev/null &
#fi

#####
# JUDGING FROM PRIMARY'S HADR STATUS FLAG
#####

if [[ "${ROLE_SNP_LOCAL}" = "Primary" ]]; then
# HADR ROLE IS PRIMARY

# already primary
echo "HADR ROLL is already PRIMARY: Exit."
logger -i -p info -t $PROGNAME "HADR ROLL is already PRIMARY: Exit."

# Start the script if the tail process is not run.
ps -ef | grep "tail -n0" | grep -v "grep"
if [[ $? != 0 ]]; then
echo "Start the script that tails db2diag.log"
logger -i -p info -t $PROGNAME "Start the script that tails db2diag.log"

#
# ${TAIL_HADR_STATUS} 2>&1 /dev/null &
#fi

elif [[ "${ROLE_SNP_LOCAL}" = "Standby" ]]; then
# HADR ROLE IS STANDBY

#####
# JUDGING START TRIGGER
#####
# TEST ORIGINAL SETTING
#START_TRIGGER="NORMAL"
#START_TRIGGER="TAKEOVER_FORCE"
#START_TRIGGER="TAKEOVER_PLANNED"

if [[ -f ${HADR_RG_START_FLAG} ]]; then
echo "Resource group is already active"
logger -i -p info -t $PROGNAME "Resource group is already active."
if [[ -f ${PLANNED_TKO_FLAG} ]]; then
echo "Start trigger is Planned takeover"
logger -i -p info -t $PROGNAME "Start trigger is Planned takeover"
START_TRIGGER="TAKEOVER_PLANNED"
else
echo "Start trigger is Unplanned takeover"
logger -i -p info -t $PROGNAME " Start trigger is Unplanned takeover"
START_TRIGGER="TAKEOVER_FORCE"
fi
else
echo "Resource group is offline"
if [[ -f ${PLANNED_TKO_FLAG} ]]; then

```

```

        echo "Start trigger is Planned Takeover"
        logger -i -p info -t $PROGNAME "Start trigger is Planned takeover"
        START_TRIGGER="TAKEOVER_PLANNED"
else
    echo "Resource group is offline"
    logger -i -p info -t $PROGNAME "Resource group is offline. Start trigger is
Normal Startup"
    START_TRIGGER="NORMAL"
    fi
fi

#####
# ACTION BY START TRIGGER
#####

case $START_TRIGGER in

NORMAL)
    # Start trigger is Normal
    echo "Start trigger is Normal: Exit."
    logger -i -p info -t $PROGNAME "HADR ROLE is STANDBY but start trigger is Normal:
Exit."
    exit 1
;;

TAKEOVER_PLANNED)
    echo "Planned takeover is executed."

    if [[ "${STATE_SNP_LOCAL}" = "Peer" ]]; then
        echo "HADR is in PEER state. Takeover(without "BY FORCE") will be executed."

        # Start the script if the tail process is not run before the planned takeover.
        # ps -ef | grep "tail -n0" | grep -v "grep"
        # if [[ $? != 0 ]]; then
        #     echo "Start the script that tails db2diag.log before planned takeover."
        #     logger -i -p info -t $PROGNAME "Start the script that tails db2diag.log before
planned takeover."
        #     ${TAIL_HADR_STATUS} 2>&1 /dev/null &
        #     fi

        logger -i -p info -t $PROGNAME "HADR is in PEER state. Takeover(without "BY
FORCE") will be executed."
        ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME}"
    elif [[ "${STATE_SNP_LOCAL}" != "Peer" ]]; then
        echo "HADR isn't in PEER state. Takeover will not be executed."
        logger -i -p info -t $PROGNAME "HADR isn't in PEER state. Takeover will not be
executed."
        RC=1
    fi
;;

TAKEOVER_FORCE)

    # In the case of non-scheduled down

```

```

    echo "In the case of non-scheduled down"

# # Check the HADR status flag.
# if [[ "${PRIMARY_HADR_STATUS}" != "0" ]]; then
#     echo "Primary hasn't been in PEER state. Execute takeover command under user's
judgement."
#     logger -i -p info -t $PROGNAME "Primary hasn't been in PEER state. Execute
takeover command under user's judgement."
#
# elif [[ "${PRIMARY_HADR_STATUS}" = "0" ]]; then
#     # try takeover by force
#     echo "Primary has been in PEER state. HADR TAKEOVER BY FORCE can be excuted
without data inconsistency."
#     logger -i -p info -t $PROGNAME "Primary has been in PEER state. HADR TAKEOVER BY
FORCE can be excuted without data inconsistency."

    ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME} by force"
    echo "db2 takeover hadr on db ${DB2HADRDBNAME} by force"

#     # Start the script if the tail process is not run.
#     ps -ef | grep "tail -n0" | grep -v "grep"
#     if [[ $? != 0 ]]; then
#         echo "Start the script that tails db2diag.log"
#         logger -i -p info -t $PROGNAME "Start the script that tails db2diag.log"
#         ${TAIL_HADR_STATUS} 2>&1 /dev/null &
#     fi
# fi
;;

*)
    # Undetermined situation
    echo "Undetermined situation"
    RC=1

;;

esac

else
    echo "HADR ROLL is neither PRIMARY or STANDBY. Check HADR ROLL."

fi

#####
# CREATE THE FLAG FILE INDICATES THAT HADR_RG STARTED
#####

# WRITE RG_START_FLAG to LOCAL NODE
echo $DATE > ${HADR_RG_START_FLAG}

# WRITE RG_START_FLAG to REMOTE NODE
${RSH} "echo $DATE > ${HADR_RG_START_FLAG}"

```



```

# REMOVE PLANNED_TAKEOVER_FLAG
rm -f ${PLANNED_TKO_FLAG}

date

exit $RC

```

---

## C.3 hadr\_stop.ksh

Example C-3 shows a sample stop script for HADR takeover in an HADR with TSA environment.

*Example: C-3 hadr\_stop.ksh*

---

```

#!/bin/ksh -x

#####
# hadr_stop.sh
# This script is called when HADR_RG stops.
# 1. Kill processes that tailing the db2diag.log.
# 2. Remove the flag file that created when HADR_RG started.
#####

PROGNAME=`basename $0`
logger -i -p info -t $PROGNAME "Starting hadr_primary_stop.sh"

exec >> /tmp/`basename $0`.log
exec 2>&1

#####
# SET PARAMETER
#####
. `dirname $0`/env

#####
# KILL PROCESSES THAT TAILS DB2DIAG.LOG
#####
kill -9 `ps -ef | grep "tail -n0" | grep -v "grep" | awk '{print $2}'`

#####
# REMOVE THE FLAG FILE
#####
#AT LOCAL NODE
ls -l ${HADR_RG_START_FLAG}

if [[ $? = 0 ]]; then
    echo "Removing ${HADR_RG_START_FLAG} at local node"
    rm ${HADR_RG_START_FLAG}
else

```

```

        echo "${HADR_RG_START_FLAG} does not exist at local node"
    fi

    #AT REMOTE NODE
    ${RSH} "ls -l ${HADR_RG_START_FLAG}"

    if [[ $? = 0 ]]; then
        echo "Removing ${HADR_RG_START_FLAG} at remote node"
        ${RSH} "rm ${HADR_RG_START_FLAG}"
    else
        echo "${HADR_RG_START_FLAG} does not exist at remote node"
    fi

    date

    exit 0

```

---

## C.4 hadr\_monitor.ksh

Example C-4 shows a sample script for monitoring the HADR state.

*Example: C-4 hadr\_monitor.ksh*

---

```

#!/usr/bin/ksh -x

#####
# hadr_monitor.sh
# This script is defined as a monitor script of HADR_RG.
#
# 1. Monitor DB2 instance, HADR role and the existence of the flag file.
# 2. Judge the HADR_RG status.
#####

PROGNAME=`basename $0`
logger -i -p info -t $PROGNAME "Starting hadr_monitor.sh"

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
echo `basename $0`
date
echo "#####"

#####
# SET PARAMETER
#####

```

```

. `dirname $0`/env
. `dirname $0`/get_hadr_info.fnc

#####
# GET DB2INSTANCE STATUS
#####
${SU_CMD} "db2gcf -i ${DB2HADRINSTANCE1} -s"
RC=$?

if [[ $RC = 0 ]]; then

#####
# GET HADR ROLE
#####

F_get_status_by_db2pd
echo ${ROLE_SNP_LOCAL}

#Check if the flag file exists
if [ -e ${HADR_RG_START_FLAG} ]; then
    echo "The flag file exists."

    #Check HADR_ROLE
    if [ "${ROLE_SNP_LOCAL}" != "" ];then

        case "${ROLE_SNP_LOCAL}" in
            Primary)#LOCALNODE is Primary
                echo "LOCALNODE is Primary and the flag file exists. RG status is
Online.(RC=1)"
                logger -i -p info -t $PROGNAME "LOCALNODE is Primary and the flag file
exists. RG status is Online.(RC=1)"

                exit 1

            ;;
            Standby)#LOCALNODE is Standby
                echo "Thouth LOCALNODE is Standby, the flag file exists. RG status is
Offline.(RC=02)"
                logger -i -p info -t $PROGNAME "Thouth LOCALNODE is Standby, the flag file
exists. RG status is Offline.(RC=2)"

            # offline RC=2
            exit 2

            ;;
            *) #LOCALNODE STATUS is unknown

                echo "No process is done."

            ;;
        esac
    else
        echo "DB2 instance is started but HADR is stopped. RG status is Stuck
Online.(RC=4)"
    fi
fi

```

```

        logger -i -p info -t $PROGNAME "DB2 instance is started but HADR is stopped.
RG status is Stuck Online.(RC=4)"
        # stuck online RC=4
        exit 4
    fi
else
    echo "The flag file doesn't exist. RG status is Offline.(RC=2)"
    logger -i -p info -t $PROGNAME "The flag file doesn't exist. RG status is
Offline.(RC=2)"

    exit 2
fi

else
    echo "DB2 instance is stopped."
    logger -i -p info -t $PROGNAME "DB2 instance is stopped."

    #Check Flag_file
    if [ -e ${HADR_RG_START_FLAG} ]; then
        echo "DB2 instance is stopped, and the flag file exists. RG status is Failed
Offline.(RC=3) HADR_RG is taken over to another node."
        logger -i -p info -t $PROGNAME "DB2 instance is stopped, and the flag file
exists. RG status is Failed Offline.(RC=3) HADR_RG is taken over to another node."

        exit 3
    else
        echo "DB2 instance is stopped, and the flag file doesn't exists. RG status is
Failed Offline.(RC=3) HADR_RG is taken over to another node."
        logger -i -p info -t $PROGNAME "DB2 instance is stopped, and the flag file
doesn't exists. RG status is Failed Offline.(RC=3) HADR_RG is taken over to another
node."

        exit 3
    fi
fi

date

```

---

## C.5 planned\_takeover.ksh

Example C-5 shows a sample script for performing the takeover in the event that a failure is detected by the monitor script.

Example: C-5 *planned\_takeover.sh.ksh*

---

```
#!/bin/ksh -x

#####
# planned_takeover.sh
# This script moves HADR_RG.
#
# 1. Create the planned takeover flag.
# 2. Move the HADR_RG.
#####

PROGRAMME=`basename $0`

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

#####
# SET PARAMETER
#####
. `dirname $0`/env
. `dirname $0`/get_hadr_info.fnc

echo "###@1:   Judge the HADR_ROLE"

#####
# GET HADR ROLE(DB2PD) ON LOCAL NODE
#####
F_get_status_by_db2pd

if [[ "${ROLE_SNP_LOCAL}" = "Primary" ]]; then
    echo "### HADR_ROLE=PRIMARY"
    # already primary
    echo "HADR ROLL is PRIMARY. This script should be executed on the standby node :
Exit."
    logger -i -p info -t $PROGRAMME "HADR ROLL is PRIMARY. This script should be executed
on the standby node : Exit."

elif [[ "${ROLE_SNP_LOCAL}" = "Standby" ]]; then
    ### HADR_ROLE IS STANDBY ###
    echo "### HADR_ROLE=STANDBY"
    date > ${PLANNED_TKO_FLAG}
    rgreg -o move $HADR_RG

fi
```

---

## C.6 get\_hadr\_info.fnc

Example C-5 shows a function for getting the HADR role and state.

*Example: C-6 get\_hadr\_info.fun file*

---

```
#!/usr/bin/ksh

#####
# get_hadr_info.fun
# This script is called by stard and monitor scripts.
#
# 1.Get HADR role and state by db2pd
# 2.Get HADR role by database configurations
#####

#####
# GET HADR ROLE & STATE BY db2pd COMMAND
#####

function F_get_status_by_db2pd {

    HADR_PROBE=`${SU_CMD} "db2pd -hadr -db ${DB2HADRDBNAME}" | head -6 | tail -n 1`
    #echo $HADR_PROBE | read ROLE_SNP_LOCAL STATE_SNP_LOCAL OTHER
    ROLE_SNP_LOCAL=$(echo ${HADR_PROBE} | cut -d ' ' -f 1)
    STATE_SNP_LOCAL=$(echo ${HADR_PROBE} | cut -d ' ' -f 2)

    if [[ "${ROLE_SNP_LOCAL}" = "" || "${STATE_SNP_LOCAL}" = "" ]];then
        echo "db2pd command failed !!!"
        return 1
    fi

    return 0
}

#####
# GET HADR ROLE & STATE BY db cfg COMMAND
#####

function F_get_role_by_cfg {

    ROLE_CFG_LOCAL=$((${SU_CMD} "db2 get db cfg for ${DB2HADRDBNAME}" | grep 'HADR
database role' | awk '{print \$5}'))

    if [[ "${ROLE_CFG_LOCAL}" = "" ]];then
        echo "db2 get db cfg command failed !!!"
        return 1
    fi

    return 0
}

```

---

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 842. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- ▶ *DB2 Integrated Cluster Environment Deployment Guide*, SG24-6376
- ▶ *Integrating IBM DB2 with the IBM System Storage N series*, SG24-7329

## Other publications

These publications are also relevant as further information sources:

### IBM - DB2 9.5

- ▶ *What's New*, SC23-5869
- ▶ *Administrative API Reference*, SC23-5842
- ▶ *Administrative Routines and Views*, SC23-5843
- ▶ *Call Level Interface Guide and Reference, Volume 1*, SC23-5844
- ▶ *Call Level Interface Guide and Reference, Volume 2*, SC23-5845
- ▶ *Command Reference*, SC23-5846
- ▶ *Data Movement Utilities Guide and Reference*, SC23-5847
- ▶ *Data Recovery and High Availability Guide and Reference*, SC23-5848
- ▶ *Data Servers, Databases, and Database Objects Guide*, SC23-5849
- ▶ *Database Security Guide*, SC23-5850
- ▶ *Developing ADO.NET and OLE DB Applications*, SC23-5851
- ▶ *Developing Embedded SQL Applications*, SC23-5852

- ▶ *Developing Java Applications*, SC23-5853
- ▶ *Developing Perl and PHP Applications*, SC23-5854
- ▶ *Developing User-Defined Routines (SQL and External)*, SC23-5855
- ▶ *Getting Started with Database Application Development*, GC23-5856
- ▶ *Getting Started with DB2 Installation and Administration on Linux and Windows*, GC23-5857
- ▶ *Internationalization Guide*, SC23-5858
- ▶ *Message Reference, Volume 1*, GI11-7855
- ▶ *Message Reference, Volume 2*, GI11-7856
- ▶ *Migration Guide*, GC23-5859
- ▶ *Net Search Extender Administration and User's Guide*, SC23-8509
- ▶ *Partitioning and Clustering Guide*, SC23-5860
- ▶ *Query Patroller Administration and User's Guide*, SC23-8507
- ▶ *Quick Beginnings for IBM Data Server Clients*, GC23-5863
- ▶ *Quick Beginnings for DB2 Servers*, GC23-5864
- ▶ *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC23-8508
- ▶ *SQL Reference, Volume 1*, SC23-5861
- ▶ *SQL Reference, Volume 2*, SC23-5862
- ▶ *System Monitor Guide and Reference*, SC23-5865
- ▶ *Troubleshooting Guide*, GI11-7857
- ▶ *Tuning Database Performance*, SC23-5867
- ▶ *Visual Explain Tutorial*, SC23-5868
- ▶ *Workload Manager Guide and Reference*, SC23-5870
- ▶ *pureXML Guide*, SC23-5871
- ▶ *XQuery Reference*, SC23-5872
- ▶ *DB2 Performance Expert for Multplatforms Installation and Configuration Guide*, SC19-1174

## **IBM - DB2 9**

- ▶ *What's New*, SC10-4253
- ▶ *Administration Guide: Implementation*, SC10-4221
- ▶ *Administration Guide: Planning*, SC10-4223



- ▶ *Administrative API Reference*, SC10-4231
- ▶ *Administrative SQL Routines and Views*, SC10-4293
- ▶ *Administration Guide for Federated Systems*, SC19-1020
- ▶ *Call Level Interface Guide and Reference, Volume 1*, SC10-4224
- ▶ *Call Level Interface Guide and Reference, Volume 2*, SC10-4225
- ▶ *Command Reference*, SC10-4226
- ▶ *Data Movement Utilities Guide and Reference*, SC10-4227
- ▶ *Data Recovery and High Availability Guide and Reference*, SC10-4228
- ▶ *Developing ADO.NET and OLE DB Applications*, SC10-4230
- ▶ *Developing Embedded SQL Applications*, SC10-4232
- ▶ *Developing Java Applications*, SC10-4233
- ▶ *Developing Perl and PHP Applications*, SC10-4234
- ▶ *Developing SQL and External Routines*, SC10-4373
- ▶ *Getting Started with Database Application Development*, SC10-4252
- ▶ *Getting Started with DB2 Installation and Administration on Linux and Windows*, GC10-4247
- ▶ *Message Reference Volume 1*, SC10-4238
- ▶ *Message Reference Volume 2*, SC10-4239
- ▶ *Migration Guide*, GC10-4237
- ▶ *Performance Guide*, SC10-4222
- ▶ *Query Patroller Administration and User's Guide*, GC10-4241
- ▶ *Quick Beginnings for DB2 Clients*, GC10-4242
- ▶ *Quick Beginnings for DB2 Servers*, GC10-4246
- ▶ *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC18-9749
- ▶ *SQL Guide*, SC10-4248
- ▶ *SQL Reference, Volume 1*, SC10-4249
- ▶ *SQL Reference, Volume 2*, SC10-4250
- ▶ *System Monitor Guide and Reference*, SC10-4251
- ▶ *Troubleshooting Guide*, GC10-4240
- ▶ *Visual Explain Tutorial*, SC10-4319
- ▶ *XML Extender Administration and Programming*, SC18-9750

- ▶ *XML Guide*, SC10-4254
- ▶ *XQuery Reference*, SC18-9796
- ▶ *DB2 Connect User's Guide*, SC10-4229
- ▶ *DB2 9 PureXML Guide*, SG24-7315
- ▶ *Quick Beginnings for DB2 Connect Personal Edition*, GC10-4244
- ▶ *Quick Beginnings for DB2 Connect Servers*, GC10-4243

### **Tivoli**

- ▶ *IBM Tivoli System Automation for Multiplatforms 2.2 Base Component Administrator's and User's Guide*, SC33-8272
- ▶ *TSM ACS V5.5 Installation and User's Guide*, SC33-8330

## **Online resources**

These Web sites and URLs are also relevant as further information sources:

### **DB2**

- ▶ DB2 Information Center  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/>
- ▶ Database and Information Management home page  
<http://www.ibm.com/software/data/>
- ▶ DB2 Technical Support  
[http://www.ibm.com/software/data/db2/support/db2\\_9/](http://www.ibm.com/software/data/db2/support/db2_9/)
- ▶ DB2 Product Family Library  
<http://www.ibm.com/software/data/db2/library/>
- ▶ DB2 developerWorks  
<http://www.ibm.com/developerworks/db2/>
- ▶ DB2 for Linux  
<http://www.ibm.com/software/data/db2/linux/>  
<http://www.ibm.com/software/data/db2/linux/validate/>
- ▶ DB2 Universal Database V9 Application Development  
<http://www.ibm.com/software/data/db2/ad/>
- ▶ Planet DB2  
<http://www.planetdb2.com/>

## **Linux**

- ▶ IBM Software for Linux

<http://www.ibm.com/software/os/linux/software/>

## **Other**

- ▶ Steven Raspudic, Selvaprabhu Arumuggharaj, and Saurabh Sehgal, *Automated Cluster Controlled HADR (High Availability Disaster Recovery) Configuration Setup using the IBM DB2 High Availability Instance Configuration Utility (db2haicu)*, IBM WhitePaper, April 2008

[ftp://ftp.software.ibm.com/software/data/pubs/papers/HADR\\_db2haicu.pdf](ftp://ftp.software.ibm.com/software/data/pubs/papers/HADR_db2haicu.pdf)

[http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/db2/papers/db2\\_db2haicu.pdf](http://download.boulder.ibm.com/ibmdl/pub/software/data/sw-library/db2/papers/db2_db2haicu.pdf)

- ▶ SAP Standard Application Benchmarks

<http://www.sap.com/solutions/benchmark/index.epx>

- ▶ DBI.perl.org

<http://dbi.perl.org>

- ▶ DB2 Perl Database Interface

<http://www.ibm.com/software/data/db2/perl>

- ▶ Comprehensive Perl Archive Network

<http://www.cpan.org>

[http://www.cpan.org/modules/by-category/07\\_Database\\_Interfaces/DBI](http://www.cpan.org/modules/by-category/07_Database_Interfaces/DBI)

- ▶ Apache HTTP Server Project

<http://httpd.apache.org>

- ▶ Perl.apache.org

<http://perl.apache.org/docs/1.0/guide/>

- ▶ PHP scripting language

<http://www.php.net/>

- ▶ IBM Tivoli System Automation for Multiplatforms

[http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8272-01/en\\_US/PDF/HALBAU01.pdf](http://publib.boulder.ibm.com/tividd/td/ITSAFL/SC33-8272-01/en_US/PDF/HALBAU01.pdf)

## How to get IBM Redbooks publications

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Symbols

.rhosts file 635

## A

about database instance 478  
access permission 173  
ACR 121, 130, 135, 139–140, 415, 443, 467, 469  
ACS 16, 721, 724–734, 736–740, 742–743, 747–748, 751–753, 755–756, 759  
ACS repository 727, 738  
ACS\_DIR 727, 736–738, 740, 754  
ACS\_REPOSITORY 727, 736, 738–739, 754  
acscim 723–724, 733, 754, 756, 758  
acsd 723  
acsdm.sh 724  
activate database 40, 376  
active backup 19  
active log 303  
active log file 765  
active mode 304  
active primary node 26, 28  
active/active 242  
active/passive pair 651  
active/standby configuration 241  
administrative view 109, 111–112  
Advanced Copy Service 16  
Advanced Copy Services 721, 731, 736  
algorithm 309  
Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) 36  
alias 509, 520, 524, 530  
alternate server 72, 122–124, 126–135, 137–140  
alternative location 122  
API 274  
application failure 16  
application resource 239  
application server 279–281, 283, 285, 291, 293–294  
application server script 291  
archive log 303, 742, 757  
archive log file 773  
archive log location 767  
Archive log mode 19

archive logging 412  
archived log 18  
ARIES 36  
ASN 682, 686, 708, 711  
ASYNCR 50, 67, 84, 89  
ASYNCR mode 147, 150, 152, 156–159, 168  
asynchronous 9  
asynchronous mode 41, 46  
asynchronous page cleaner 305  
authorized\_keys file 420–422  
automatic client reroute 34, 53, 121–123, 125–126, 128, 130–132, 134, 139–140, 177, 349, 352, 374, 380, 415, 443, 659  
automation adapter 235  
automation component 235  
automation layer 235  
automation policy 237–238  
AUTORESTART 90  
availability 1–2, 8, 10, 14, 18–20, 25–26

## B

background process 7  
backup 229, 720–722, 724–729, 732, 738–746, 748, 754, 756–757, 759  
backup command 714  
backup database command 766, 770, 800  
backup file 759  
backup files 94  
backup hosts 726  
backup image 61, 714, 716, 719  
backup path 409  
backup repository 723  
backup timestamp 714, 719  
base code 194  
base component 235  
base-level cluster 235  
basic operation 74  
batch job 306  
batch mode 480  
bidirectional 24  
bind commands 187  
binder 237  
BLOB 101

- bonded network features 461
- broadcast address 248
- buffer 722
- buffer overflow 419
- buffer pool 36–38, 48, 100, 305
- buffer size 150
- business requirements 16
- BY FORCE option 74, 77, 83–84

## C

- capture program 22
- case-sensitive 479
- catalog 628
- catalog node 714, 760
- catalog partition 628, 720, 723, 743–744
- catalog table space 303
- catch-up phase 40, 42
- category 726–727
- chcomg 261
- chrg 269, 392–393, 400, 433–435, 439–440, 452, 454–455, 457, 466, 473
- chrsrc 388, 429
- CIM 723, 725, 729, 731, 738, 752–753, 756, 758
- CIMOM 752, 755–756
- circular logging 49, 55–56
- client access 483, 485, 581
- client machine 137
- client reroute 121–126, 128–132, 134–135, 138–140
- client service 279
- CLOB 101
- CLP 50
- cluster 10–13, 25–28, 307–312, 315–316, 318–321, 323–324, 327–329, 331–332, 334, 336–342
- cluster definition 295
- cluster domain 478, 481, 483, 485, 487–490, 493, 495, 510, 514, 517, 540–541, 543–545, 547–548, 552–557, 567–569, 571, 573, 581, 585, 588, 591, 593, 595, 605, 608, 610, 625–626, 651
- cluster environment 627
- cluster management 408
- cluster manager 272–274, 406–408, 478, 489, 513, 516, 518, 521, 524–530, 533, 535–537, 570–571, 574, 580
- cluster node 407, 410, 428–429, 437, 443, 461, 482, 636, 651
- cluster pair 407

- cluster server concept 308
- clustered environment 478
- clustering 9–13, 20
- clustering solution 10
- CMVWL 274
- command line 47, 67–68, 71–72, 74, 77, 88, 94, 97
- Command Line Processor 50, 177
- command line step 94
- commit point 775
- commit request 154, 164
- commit statement 152
- common information model 723
- communicate 272
- communication 738
- communication error 123
- communication interface 282
- communication port 229
- component 273, 723–724, 752
- compress 722
- compression 154
- compression library parameter 768
- configuration 49, 51–54, 56, 61, 67–68, 72–73, 84, 92, 273
  - target to source server connectivity 707
  - using Setup HADR wizard 52
- Configuration Assistant 60
- congestion 150, 152, 156–158
- connection 272
- connection attempt 137
- connection interval 123
- connection status 105
- consistent PIT 766
- context object 136
- contingency plan 14
- continuous rollforward mode 34
- Control Center 53–54, 58, 60, 70–71, 75, 77–78, 80, 85, 87–88, 97, 177
- coordinator 723
- copy image 17
- COPYSERVICES\_COMMPROTOCOL 754, 756
- COPYSERVICES\_HARDWARE\_TYPE 736, 738–739, 754–756
- COPYSERVICES\_PRIMARY\_SERVERNAME 736, 738–739, 754–755
- COPYSERVICES\_SERVERPORT 754, 756
- COPYSERVICES\_TIMEOUT 754, 756
- COPYSERVICES\_USERNAME 736, 738–739, 754, 756
- crash recovery 291, 304

- creating a cluster server 312
- creating a DB2 group 331
- creating a DB2 instance 327
- critical failure 656
- Critical resource 236
- critical service 280
- CT\_MANAGEMENT\_SCOPE 590
- CURRENT\_TIMESTAMP 768
- cursor 128, 140
- custom compression library 768
- custom library 768

**D**

- daemon 410, 419–420, 434, 630
- dark fiber 8
- data backup 14
- data consistency 15
- data corruption 407
- data definition language (DDL) 100
- data integrity 794
- data links 101
- data loss 14, 84–85
- data manipulation language (DML) 100
- data set 720
- data source attributes 136
- data source movement 33
- data storage 302
- database
  - backup 101, 103
  - restore 101, 103
- database backup 714, 739, 742, 744
- database component 279–280
- database configuration 101, 104
- database configuration parameter
  - softmax 305
- database configuration parameters
  - AUTO\_DEL\_REC\_OBJ 793
  - autorestart 145
  - chnpggs\_thresh 305
  - dftdbpath 409
  - hadr\_db\_role 145
  - hadr\_local\_host 146, 409, 509
  - hadr\_local\_svc 146, 409–410, 509
  - hadr\_peer\_window 147, 479, 511
  - hadr\_remote\_host 146, 409, 509
  - hadr\_remote\_inst 146, 409, 509
  - hadr\_remote\_svc 146, 409, 509
  - hadr\_syncmode 146, 409
  - hadr\_timeout 147, 409, 509
  - indexrec 148
  - logarchmeth1 68, 409
  - logarchmethn 145
  - logfilsiz 148
  - logindexbuild 145, 168
  - logretain 72, 145
  - NUM\_DB\_BACKUPS 763, 773
  - num\_iocleaners 305
  - peer\_window 395
  - self\_tuning\_mem 148
  - sysadm\_group 87
- database failure 766
- database instances 273
- database logs 17, 714
- database managed space 302
- database manager 272, 478
- database manager configuration parameters
  - max\_time\_diff 629
  - svcename 603
- database manager instance 480
- database name 216
- database operations 167
- database pairs 478
- database partition 272, 479–480, 629, 715, 737, 745, 755, 757, 759
- database partition server 637
- database path 721
- database recovery strategy 18
- database restore 714
- database role 497, 509–510, 528
- database server partition 637
- datasource 126, 136, 138
- datasource object 136
- DB parameter 49
- db\_data 726
- db\_log 726
- DB2
  - configuration parameter 231
  - FixPak rolling upgrade 177
  - system upgrade 175
  - version upgrade 194
- DB2 Admin Server (DAS) 53
- DB2 engine 38
- DB2 FixPak
  - using command line 191
- DB2 HADR 32, 34, 39
- DB2 logical log architecture 36
- db2 registry variable

- db2\_connretries\_interval 125
- db2\_hadr\_buf\_size 149
- db2\_max\_client\_connretries 125
- db2loadrec 151
- db2tcp\_client\_contimeout 125
- load\_copy\_no\_override 150
- DB2 with Microsoft Windows Cluster Server 307
- db2\_all 714, 745–746
- DB2\_CONNRETRIES\_INTERVAL 126
- DB2\_GRP\_LOOKUP 87
- db2\_install 276
- db2\_kill 292
- DB2\_MAX\_CLIENT\_CONNRETRIES 126
- db2acsutil 723–724, 740, 744–745, 758
- db2acsutil command 792
- db2admin 227
- db2adutl command 792
- db2agent 37–38
- db2ckbkp 715
- db2ckmig 218
- db2ckmig command 194
- db2clini.ini 126
- db2cptsa 276
- db2diag.log 375, 395, 426, 776
- db2diag.log filtering output 118
- db2fm 603
- db2gcf 296, 298, 413–414, 424, 426, 434
- db2hadrp 39–40
- db2hadrs 39–40
- db2haicu 168, 272–273, 478–479, 666
- db2havend 274
- db2host 299
- db2icrt 327, 510, 541–542, 566, 603
- db2ilist 327, 336–337
- db2inidb 721
- db2level 227
- db2lfr 38, 40
- db2loggr 38–39
- db2loggw 38
- db2nodes.cfg 272, 277, 291, 295–300, 637
- DB2PartitionSet 652
- db2pd 71, 88–89, 96–97, 372, 374–376, 395, 398, 400, 408, 412, 414, 437, 442, 446, 448–451, 455–458, 472–475, 647
- db2profile 478
- db2Prune 17
- db2start 272, 478
- db2start command 74
- db2stop 292, 665
- db2support command 217
- DB2TCP\_CLIENT\_CONTIMEOUT 126
- DB2TCP\_CLIENT\_RCVTIMEOUT 125
- db2updv9 193
- DBM CFG parameter 49
- dbpartitionnum parameter 480
- DBPATH 721
- DDoS 419
- deactivate database 371, 377
- deactivate database command 77, 197
- default partition group 761
- delete resource groups 480
- dependency 378, 406–407, 470, 479
- dependency relationship 433–434
- desired state 237
- device drivers 729
- device file 249
- device files 251
- device volume 724
- diagnostic log 482
- direct access mode 235
- dirty page 305
- disaster recovery 1, 9, 14, 18, 21, 26
- disaster recovery plan 14
- disaster recovery procedure 15
- disaster recovery solution 14, 25
- disaster recovery tier levels 14
- DisconnectedPeer 105
- disk array 8
- disk controller 3, 663
- Disk Mapper 724
- disk mapper 724
- disk mapping information 724
- disk mirroring 3–4
- disk resources 279
- disk space 764
- disk striping 3
- distributed denial of service 419
- distributed management task force 723
- dll 310
- DMS table space 759
- DMTF 723
- DNS 50, 59, 89, 299
- domain 311–316, 323, 327
- domain name 639
- domain name server 299
- domain named 407
- domain node 407
- downloadable package 194



DS6000 722–724, 728–730, 738, 752, 756  
DS8000 722–724, 728–730, 738, 752, 756  
dummy host 726, 749–750  
dummy script 283  
Dynamic Link Library (dll) 310

## E

EID 797  
element 651  
embedded loop 760  
end-to-end automation adapter 235  
engine dispatch unit (EDU) 38  
enhanced concurrent volume group 282, 303  
entry identifier 797  
equivalency 238, 245, 266–267, 385, 390, 407,  
409, 427, 430–431, 443, 459–460, 464, 479  
equivalency definition 459–460  
equivalency resource 406, 432  
error code 652  
ESS800 723, 738, 756  
ethernet 284, 301  
exclusive mode 786  
explicit path qualifier 434  
exportvg 251  
extension 723  
external communication 406

## F

failback 20, 35, 43, 45–46  
failed offline state 654–655  
failover 25, 32–33, 35, 39, 43–44, 46, 233, 281,  
346–353, 371, 373–375, 378–379, 382, 387,  
394–396, 401–402, 404, 432, 452, 461, 466, 470,  
473, 483, 489–491, 501–502, 504–505, 507–508,  
516–518, 540, 543–545, 547, 551, 554–556, 559,  
571, 580–581, 583, 585, 589, 591–592, 595, 600,  
604–606, 610, 612–613, 615, 618–619, 622–623,  
636  
failover node 251  
failover pair 638  
failover policy 643, 651  
failure 32–33, 36, 38–39, 43–44, 46, 272  
failure detection 33  
fast communications manager 603  
fault monitor 479, 603, 635  
FCM 581, 593, 603, 606, 616–618, 620, 628  
federated processing 128  
fiber-optic cable 664

file permission 635  
file system 61, 278, 286, 406, 408, 410, 633, 665  
FlashCopy 7–8, 16, 171, 720–721, 723, 726, 738,  
748, 756–757, 759  
FlashCopy relationship 7  
flush 154  
force 39–40  
force applications 216  
framework 234  
FS 286  
fsck 302, 304, 633  
FTP 61  
full database backup 18  
full offline backup 766  
full volume copies 7

## G

gateway 409  
gateway IP address 429  
gateway tiebreaker 509  
general parallel file system 631  
get db cfg 89  
GET SNAPSHOT FOR DATABASE command 103  
get snapshot for db 88–89, 96  
getstatus 433–434, 436, 438–439, 444–445  
global temporary table 128  
global variables 425  
GPFS 631  
Graphical User Interface 177  
Graphical User Interface (GUI) 47  
groups 311  
GUI 177  
GUI HADR Setup wizard 50

## H

HA clustering software (HACMP) 32  
HA policy 276  
HACMP clusters 278  
HACMP takeover 724  
HADBSet 652  
HADR ASYNC mode 473  
HADR log transfer 483  
HADR NEARSYNC mode 473  
HADR SYNCMODE 50  
HADR\_TIMEOUT 39, 45  
hadrdp parameter 480  
hardware 479  
heartbeat 35, 40, 106, 109, 152

- heartbeat device 282
- heart-beating disk 282
- high availability 1–2, 26, 272
- high availability (HA)
  - continuous availability 33
  - failover availability 33
  - overview 32
  - types 33
- high availability cluster multi-processing 277, 295
- High Availability Disaster Recovery (HADR) 31–32
  - administrating systems 100
  - administration and monitoring 99
  - architecture 39
  - benefits 35
  - communication 35
  - configuration 49
  - considerations 90
  - monitoring
    - administrative view and table function 109
    - db2diag 113
    - db2pd 106
  - overview 33
  - re-establishing after failure 92
  - replication 40
  - setup 47
  - shutdown 74
  - startup 74
  - synchronization modes 41–42
  - topology 40
- High Availability Instance Configuration Utility 478
- high-capacity network 49
- highly available environment 278
- home directory 250, 291, 295, 299, 422, 727, 734, 737
- host 726–727, 735, 748, 755
- host name 132, 272, 636, 639, 735, 737, 755
- hosts.equiv 297
- hot site 14
- hot standby mode 416
- hybrid 25–27, 29

**I**

- ICMP 407
- identity 128
- ifdown 459
- implicit path 434
- importvg 250–252
- inactive 43

- INCLUDE LOGS 714
- incremental 722, 728
- index 169
- indexing information 773
- indoubt 145
- inflight 145
- in-flight transaction 151
- infrastructure 272
- initial setup 666
- input file 273
- input value 639
- install media 273
- installation log 275
- installer 273
- installing DB2 ESE 323
- installSAM 252, 274
- instance 272, 278–280, 284–285, 290–291, 295–300, 302–304, 307, 311–312, 323–325, 327–329, 331, 334, 336–343
- instance configuration 272
- instance home directory 635–636, 737
- instance level 635
- instance node name 57
- instance owner 478, 647
- Instance resource group 468
- instance resource group 665
- instance-owning database 636
- integrity check 302, 304
- interactive mode 480
- internet control message protocol 407
- inter-partition communication 628
- IP address 234, 236, 238, 245, 248, 255, 262, 267, 279, 310, 316, 324, 326, 329, 334–335, 340
- IP address redirection 34

**J**

- java.sql.DriverManager 134
- javax.sql.ConnectionPoolDataSource 134
- javax.sql.DataSource 134
- javax.sql.XADataSource 134
- JDBC 124
- JFS 286–287
- JFS2 286–288
- JFS2LOG 286
- JFSLog 286
- JNDI 135–138
- Journalled file system 286

## K

keepalive 280, 284–285, 295, 469–470  
kernel 155  
kernel parameter 628, 632  
keyword 276  
ksh 275

## L

LEAD 50–53, 63, 71–73, 94, 96–97  
libacsd2 723, 740  
libacsd2.so 723  
license key 291  
linear/archival logging 51, 55, 61, 68  
list application 216  
list history command 773, 797  
list tablespaces command 785  
liveness check 387  
load balancing 461  
LOAD COPY NO 102  
LOAD COPY YES 102–103  
LOAD utility 172  
local Administrators group 179  
local area network (LAN) 51  
local directory 734, 740  
local HA solution 25  
local instance 409  
local quorums 309  
local registry 87  
local standby node 26  
local user ID 87  
local user manager 179  
location relationship 237  
lock state 665  
lock status 496  
log archive 174, 727  
log archive device 174  
log buffer 39  
log chain 765  
log control file 766  
log file 22, 145, 149, 154, 172, 275  
log file management 765  
log file reader (LFR) 38  
log file size 49  
log files 103, 714–715  
log gap running average 106, 109  
log index build 169  
log position 104, 106  
log reader 38

log record 36, 38  
log records 34, 765  
log sequence number (LSN) 103  
log shipping mode 44, 151–152, 154  
log shipping solution 19  
logic deck 237  
logical data group 109, 111  
logical log 36, 38, 40  
logical log buffer 38  
logical log buffer (llb) 36–37  
logical log file 36  
logical log pages 40  
logical log record (LLR) 36  
logical name 136  
logical partition 628  
logical unit of work 36  
logical volume 250, 286, 288–289, 720, 751  
logical-port parameter 272  
LOGINDEXBUILD 68, 72, 169  
LOGTARGET 716  
lowercase 479  
lsequ 390, 427, 430, 460–461  
lsrg 265  
lsrsrc 264  
lssam 492–493, 496–502, 505–508, 518–519, 521, 524–529, 533–535, 548, 551–552, 554–558, 573–580, 599, 601, 608–609, 611, 613–614, 616–619, 621, 625–626, 646, 656  
lvlstmajor 249, 286

## M

main script 279  
maintenance mode 481, 515  
major device number 249  
major number 286, 289  
majority node set (MNS) 309  
manageability 303  
managed cluster 406  
management agent 723, 737  
management script 425  
manager vendor wrapper library 274  
manually configuring DB2 instance in Windows Cluster 329  
master console 738, 747, 750–751  
master password 737  
master server 737  
MAX\_VERSIONS 736, 738–739, 754  
maxRetriesForClientReroute 126

- member resource 493
- message broker 21
- message queue 21, 24
- methodologies 14
- Microsoft Cluster Server definitions 310
- Microsoft Cluster Servers (MSCS) 308
- Microsoft Management Console Services Snap-in 186
- migrate database command 218
- migrating the DB2 instance to the cluster environment 336
- migration fallback 216
- MINCOMMIT 38
- minimum configuration and sample setup 311
- minimum required resource 406
- minimum rollforward PIT 766
- mirroring 2–4, 6, 8–9, 16
- mkequ 390, 430–431
- mkrsrc 388–390, 429
- modular 279
- monitor element 108–111, 113, 120
- monitor element output 107
- monitor script 239–240, 259
- monitoring
  - administrative view and table function 109
  - db2diag 113
  - db2pd 106
  - summary 119
- monitoring local resources 407
- mount point 654
- mounted file systems 408
- MQSC command line 678
- multi-level backup 8
- multi-partitioned database 627, 714
- multiple instances 737
- multiple network adapters 406, 431
- multiple parallel chain 800
- mutual takeover 242, 408

## N

- name resolution 285
- native data format 794
- NEAR SYNCHRONOUS mode 41
- near-readiness 34
- NEARSYNC 50–51, 67, 84, 89
- near-sync mode 46
- NetBIOS 59
- netmask 248

- netmon.cf 389, 429–430
- netmon.cf file 260
- netname 637
- network adapter 236, 390, 406, 410, 416–417, 432, 459, 461
- network bandwidth 50
- network card 146, 483, 581
- network communication 406
- Network configuration 628
- network configuration 284
- network equivalency 245, 266
- network failure 661
- network file system 173
- network interface 245, 266, 432, 651
- network interface card 234, 416
- network load balance 308
- network object 416
- network parameter 125
- network quorum 640
- network time protocol 482, 628
- network topology 444
- NEWLOGPATH 757
- NFS 173
- NFS mount 727, 755
- NFS server 727
- nfs\_disk 727
- NIC 402, 416–418, 430–432, 459, 461, 464
- NLB 308–309
- noauto 633
- node 10–11, 26–28, 478
- node crash 655
- node number 637
- no-downtime benefit 175
- nominal operational state 408
- nominal state 268–269, 448, 450, 461, 463, 466
- non logged large object 101
- non-catalog node 655
- nonrecoverable load 102
- non-replicated operation 101
- notify log 776
- NTP 482, 628
- NUM\_DB\_BACKUPS 801

## O

- object container 234
- object tree 180, 216
- observed state 237
- offline 94, 654

offline backup 772  
offline REORG 100  
off-site facility 16  
OLTP 5–7  
ON ALL DBPARTITIONNUMS 714–715  
on all dbpartitionnums 764  
ON DBPARTITIONNUM 714  
online 94, 272, 654  
online backup 412, 742  
online operational state 408  
online REORG 100  
operating system (OS) 231  
operating system library 19  
OpState 268  
original primary 177  
out of band 44  
outage period 44  
outbound public key 422  
overflow log path 719  
overhead 764

## P

p\_disk 726  
packet 155  
page cleaner 305  
parallelism 722  
parameter 479  
partition server 636  
partition server number 637  
partition servers 629  
partitioned database 272, 479  
partitioned database backups 764  
partitioned environment 723  
passive node 26  
password 635  
password file 737, 755  
password-less SSH 420, 423  
peer domain 236, 261, 385, 387–388, 406, 411, 427–429, 441, 478  
peer domain scope 590  
Peer state 35, 40–41, 44, 46, 49, 66, 71, 79, 83–84, 88, 92, 95–97, 100, 104, 106, 113, 116–118, 353, 355, 372, 375–376, 382, 403, 405, 412, 437, 439, 442, 448, 451, 455, 458, 472–474  
peer to peer 24  
peer window 163–167  
peer-domain 407  
PeerWindow 107–109  
PeerWindowEnd 105, 107–109  
pending offline 464  
pending online 464  
pending online state 496, 526, 580  
performance 2–4, 6–10, 20–21  
permissions 360, 410, 426, 434  
physical device 236  
physical IP address 587  
physical network adapter 406, 416  
physical node 284  
Physical nodes configuration 627  
physically copied 7  
PhysicalNetwork 651  
pickup truck access method 15  
Planned takeover 79  
planned takeover 79, 83  
point in time 720  
point-in-time copy 7, 170  
point-in-time recovery 714  
policy script 276  
POLONIUM 51–53, 57, 60–61, 63, 71–73, 80–81, 85, 94–95, 97  
port 39  
port number 135, 146, 149  
port range 635  
portNumber 134  
post-installation 194  
power interruption 16  
preprnode 488, 511–512, 544, 567, 591  
prereqSAM 275  
primary 33–35, 39–46, 483–486, 490–493, 496–508, 510, 517–518, 522, 525, 528, 536–537, 569, 585, 623–624  
primary cluster pair 302  
primary database 20, 26, 34–35, 39, 44–45, 61, 74, 94, 346, 348–349, 351–355, 369, 372, 374–379, 382, 398, 400, 402–403, 405, 407  
primary database reintegration 85  
primary database server 34  
primary node 483, 490, 492, 497–508, 525, 528, 585  
primary reintegration 45  
Primary role 475  
primary role 404, 406–408, 439, 442, 444–445, 447–448, 467–468, 474  
primary server 346, 348–349, 412  
primary system 33  
private network 628  
profile 736–738, 754

property 126–127, 135, 138, 140  
protocol 127, 756  
proxy 407  
PRUNE HISTORY 17  
prune history command 794, 805  
PTAM 15  
public key-pair 421  
Public network 628  
public network 248  
pwd.acsd 737, 755

## Q

Q Apply Control table  
  creation 685  
Q Capture Control table 674  
  creation 680  
Q Capture program 674  
Q replication 23, 673–676, 680, 685, 689, 691, 707  
  bidirectional 674  
  introduction 674  
  peer-to-peer 674  
  types 674  
  unidirectional 674  
Q subscription 674–675  
  creation 689  
Q-Replication 2, 21  
quiesce instance command 785  
quiesce tablespaces for table command 785  
quorum 236, 260, 262, 309–311, 320, 324, 407,  
489, 491–492, 504, 515, 545, 548, 569, 608, 622  
quorum cluster server 308–309  
quorum node 407

## R

RAID 2–8  
RAID 0 3  
RAID controller 3  
raw device 278, 302–303  
raw devices 278  
rc.db2pe 292  
REC\_HIS\_RETENTN 792, 801  
receiving socket 155  
recfgct 513  
recover database command 759, 770, 773  
recovery 1, 9, 14–18, 21, 23, 26  
recovery chain 783  
recovery event 765  
recovery history file 101–102, 759, 765, 773, 785

recovery point objective 18  
recovery time 14  
recovery time objective 8, 18  
Redbooks Web site 842  
  Contact us xiv  
redundancy 3, 5, 29  
redundant 122  
redundant array of independent disks 2–3  
regdb2salin 425–427, 432–434, 437, 465  
registration command 437  
registry variable 125, 149–151, 158, 412, 424  
registry variables 125  
  DB2\_NUM\_FAILOVER\_NODES 479, 606  
relational locking 794  
relationship type 391  
Reliable Scalable Cluster Technology 275  
remote catchup 104, 118–119  
remote catchup pending 105  
remote client requests 406  
remote DB alias 409  
remote shell 297, 628  
remote site 35  
replicating LOAD operations 102  
replication architecture 22  
replication center 709, 711  
replication solution 21, 23  
repository 135, 138  
reserved port 410  
resource 278, 307, 309–310, 315, 324–332,  
334–336, 338–340, 342–343, 479  
resource dependencies 407  
resource failures 407  
resource group 27, 133, 234, 241, 245, 265,  
267–272, 278, 280–283, 285, 292–294, 301–303,  
352, 358–359, 361–362, 364–365, 367–369,  
372–376, 380, 385, 391, 393–397, 399–400, 402,  
404, 406–408, 432–434, 437, 439–440, 444–445,  
447, 452, 455, 457, 459, 464, 468, 473, 654, 665  
resource group operational state 408  
resource groups 294  
resource monitoring and control 512  
resource status  
  offline 406  
  online 406  
resource type 310, 329–331  
response file 276  
restore 16, 219, 720–721, 723–724, 728, 739, 741,  
743, 745–746, 756, 759  
restore command 717, 721

- restore database command 716
- restore operations 17
- retry logic 177
- retryIntervalForClientReroute 126
- return code 239–240
- risk 14
- RMC 512
- role 104, 117
- rollforward 227, 721
- rollforward database command 719, 759
- rollforward pending mode 412
- rollforward pending state 187
- rollforward recovery 17, 719, 721
- rolling upgrade 20, 35, 177, 191
- root authority 478
- RSCT 275, 498, 512–513, 552, 584
- rsh 297–298, 410, 419–420, 425, 628, 635
- RTO 8, 18, 20
- runtime 273

**S**

- SA MP 273
- sacrificed state 237
- sa-lvm script 256
- schedule 18–19
- scripts 274
- SCSI 309, 315
- SCSI disk 237
- Secure Shell 410, 419–420
- secure shell 297
- security 478
- security policy 87
- send queue 674
- sequence 275
- sequences 128
- serial network 295
- serial network connection 285
- serial number 756
- serial numbers 724
- server node 302
- server option 128
- serverName 134
- service 729, 732, 752, 755–756
- service address 278, 284–285, 299, 301
- service IP 132, 282, 483, 485, 497, 505, 518, 520, 540, 555, 581, 593, 595, 599
- service IP address 367, 406
- service level 14

- service name 146
- service node 278–281, 284–286, 290, 295, 483–484
- services 279
- session resource 128
- set write suspend for database 721
- setup
  - initial database source and target 676
  - preparing the environment 49
  - recommendations 48–49
  - requirements 48
  - WebSphere MQ objects on source server 676
  - WebSphere MQ objects on target server 677
- Setup wizard 275
- shared devices 309
- shared directory 635
- shared disk 279, 281–282, 284–285, 290, 292, 295, 302, 309–312, 315, 318, 324, 329, 332, 340, 406–407
- shared disks 281
- shared volume groups 278
- shared-nothing 408
- shell script 724
- shutdown 74, 77–78, 91
  - from the Control Center 78
  - using command 77
- single partition 291, 295
- single partition environment 734
- single quorum servers 309
- single system view 714, 744
- single system view backup 714
- site data 14
- sleep time 125
- SMS 759
- Snap-In 179
- snapshot 99, 103–104, 109, 113, 117, 119, 170–171
- snapshot backup 16, 721, 723–725, 728, 740–742, 748, 756–757
- snapshot capable device 724
- snapshot function 720
- snapshot restore 724
- snapshot table function 99
- soft checkpoint 305
- software prerequisite 729–730
- software requirement 483, 581, 584
- software-based replication 21
- source 721, 726, 747–748, 756
- source system 22

- source volume 7, 748, 756
- spare disk 3
- split brain 45–46, 84, 91–92, 465–467, 472–473
- SQL query 109–110
- SQL replication 2, 21–23, 25, 32
- SQL1018N error 58–59
- SQL30081N Communication error 89
- SQLJ 124
- sqllib 272
- SQLLOGCTL.LFH 766
- ssh 297–300, 384, 410, 419–420, 423–424, 427
- SSV 714
- SSV backup 714, 717
- SSV backup image 714
- staging environment 29
- standard database 45, 764
- standard mode 45
- standby 302, 484–486, 488, 490–493, 496–499, 501, 503–508, 510, 517, 524, 528, 533–534, 557, 559, 585, 593
- standby connection 39
- standby database 19–20, 26, 28–29, 34, 39–40, 45, 74, 478
- standby node 279–281, 284, 289, 295, 298, 301–302, 304
- standby role 242, 371, 405, 408, 439
- standby server 34, 229
- standby system 177
- start hadr 353, 371, 376, 403–405, 414
- START HADR AS PRIMARY 103
- START HADR command 39, 74, 77, 85
- starting listener and channel on source and target 678
- starting Q Apply 710–711
- starting Q Capture 708–709
- startup 74
- Startup mode 481
- startup mode 517
- state 100, 104, 106, 113–120, 273
- static equivalency 390
- static IP addresses 247
- STOP HADR 47, 77–78
- stop scripts 234
- Stop-start relationships 237
- storage 721, 723, 725–730, 734–735, 740, 747, 749, 752, 756
- Storage Area Network (SAN) 309
- storage capacity 7
- storage copy 171

- storage device 8, 26–28, 721, 727
- storage enclosure 8
- storage failure 16
- storage pool 727
- storage server 729, 734–735
- storage space 17–18
- storage subsystem 720–721, 729, 756
- stored procedure 19
- stored procedures and UDF 100
- structural conversion 216
- su 517, 521, 542, 544
- subdirectory 410, 412, 421, 425–426, 434, 465
- sub-element 651
- Subnet mask 409
- subsystem 657
- SUSE Linux 10 49
- SVC 723, 726, 728, 730, 738, 747–748, 750–754, 756–757
- SVC\_COPY\_RATE 754, 756
- SVCENAME 256, 637
- switch role 96
- sync mode 46
- synchronization mode 150, 154, 169
- synchronous 9, 21
- synchronous mode 41, 46
- system clocks 629
- system security 51
- System z 274

## T

- table function 109–110, 112, 117, 119–120
- table space 16, 100, 102, 303
- table space container 37, 759
- takeover 32, 34–35, 45–46, 242
- takeover by force 83–85, 92
  - GUI 85
- takeover command 79, 91, 96, 497, 521, 532
- takeover hadr command 79
- takeover mode 281
- tape-based solution 15
- tapes 16
- target directory 716–718
- target service level 14
- target system 21
- target volume 7, 726, 748–749, 751, 756
- target volumes 734, 755
- TCP/IP communication 634
- TCP/IP interface 49



tcp\_keepinit 125  
tcp\_recvspace 155  
temporary table spaces 303  
terminate 292  
terminology 32, 42  
testing the queues 679  
text based utility 273, 478  
thread 37, 46  
tiebreaker 236, 244, 378, 380, 388, 406–407, 409,  
418, 429, 465, 483, 485, 509, 539, 559, 569, 581,  
583, 640  
tier 0 14  
timeout 167, 426  
timestamp 773  
Tivoli System Automation (TSA) 13, 32, 153, 282,  
483  
tmcscli 301  
token 301  
topology 40, 278–280, 283, 295  
trace information 737  
transaction 32–33, 35–36, 38, 42, 44, 46  
transaction integrity 15  
transaction log 19  
transactional integrity 36  
transactions 16  
tree structure 60  
troubleshooting 47, 59, 86  
    after HADR disconnect or server failure 90  
    after setup or during normal execution 88  
    during setup 87  
TSA 345–347, 377–381, 383–386, 389, 391,  
393–398, 400–402, 404–407, 409–411, 416,  
419–420, 425–427, 429, 432–434, 437, 439, 442,  
444, 448, 450–451, 461, 465–466, 468–469, 475  
TSA domain 235  
TSM ACS 723  
Type 2 134  
Type 4 134  
Type 4 connectivity 126  
types of cluster 308

## U

unidirection Q replication  
    setup 675  
unidirectional 23–24  
unidirectional setup 675  
uninstallSAM 274  
unique names 287

unique public key 420  
units of work 36  
update history command 797  
upgrade 273  
usability 723–724  
user data 628  
user table space 303  
using GUI 80  
util\_impact\_priority 722  
utility 272

## V

varyon 171, 304  
verification utility 295  
version recovery 16  
version upgrade 175, 194, 224  
    Windows 196  
virtual disk 747  
virtual IP 483, 485, 490, 497, 501, 504, 507–508,  
517–518, 522, 524, 533, 545, 572, 581, 583, 616,  
619–620  
virtual IP address 384, 389, 393, 406, 432, 587,  
628, 654  
virtual machine 10  
Volume group 286  
volume group 171, 269, 280, 286, 290, 724, 751  
Volume Group Takeover utility 724  
volume pair 7  
volume size 756  
VOLUMES\_DIR 727, 754–755

## W

WebSphere MQ messages queues 674  
WebSphere Replication Server Q Replication 32  
Windows version upgrade 196  
wireless area network (WAN) 50  
with force option parameter 794  
workload 10, 308  
write ahead logging (WAL) protocol 36  
write I/O load 305

## X

X11 emulator 50  
XML file setup mode 488, 605  
XML input file 480





**Redbooks**

# High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows

(1.5" spine)  
1.5" <-> 1.998"  
789 <-> 1051 pages







# High Availability and Disaster Recovery Options for DB2 on Linux, UNIX, and Windows



**Learn HADR setup, administration, monitoring, and best practices**

**Use HACMP, MSCS, TSA with DB2 and DB2 HADR**

**Protect data with DB2 disaster recovery options**

As organizations strive to do more with less, IBM DB2 for Linux, UNIX, and Windows provides various built-in high availability features. DB2 further provides high availability solutions by leveraging enterprise system resources with broad support for clustering software such as HACMP, TSA, and Microsoft Windows Cluster Server.

This IBM Redbooks publication describes DB2's high availability functions and features, focusing on High Availability Disaster Recovery (HADR) in the OLTP environment. The book provides a detailed discussion of HADR, including setup, configuration, administration, monitoring, and best practices.

We explain how to configure cluster software HACMP, TSA, and MSCS with DB2 and show how to use these products to automate HADR takeover.

DB2 also provides unprecedented enterprise-class disaster recovery capability. This book covers single system view backup, backup and restore with snapshot backup, as well as the db2recovery command, in detail.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

### **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)