

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

Learn DB2 HADR setup, administration, monitoring, and preferred practices

Use PowerHA, MSWFC, Tivoli SA MP with DB2, and DB2 HADR

Protect data with DB2 disaster recovery options

Stanislaw Bartkowski
Ciaran De Buitlear
Adrian Kalicki
Michael Loster
Marcin Marczewski

Anas Mosaad
Jan Nelken
Mohamed Soliman
Klaus Subtil
Marko Vrhovnik
Karol Zimnol

Redbooks



International Technical Support Organization

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

October 2012

Note: Before using this information and the product it supports, read the information in "Notices" on page xi.

Third Edition (October 2012)

This edition applies to DB2 for Linux, UNIX, and Windows Version 10.1.

© Copyright International Business Machines Corporation 2007, 2012. 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

Noticesxi
Trademarks	xii
Preface	xiii
The team who wrote this book	xiii
Now you can become a published author, too!	xvi
Comments welcome.....	xvii
Stay connected to IBM Redbooks	xvii
Summary of changes.....	xix
October 2012, Third Edition	xix
Chapter 1. DB2 high availability and disaster recovery overview	1
1.1 Introduction	2
1.1.1 High availability	2
1.1.2 Disaster recovery	5
1.2 High availability solutions with DB2.....	6
1.2.1 High Availability Disaster Recovery (HADR).....	6
1.2.2 DB2 high availability (HA) feature	7
1.2.3 High availability through disk mirroring	10
1.2.4 High availability through log shipping	10
1.2.5 Automatic client reroute	11
1.3 Disaster recovery solutions with DB2	11
1.3.1 Backup and recovery options	11
1.3.2 High Availability Disaster Recovery (HADR).....	13
1.3.3 Replication	14
1.3.4 InfoSphere Change Data Capture (CDC)	17
1.3.5 Remote disk mirroring.....	18
Chapter 2. DB2 with IBM Tivoli System Automation for Multiplatforms .	19
2.1 Overview	20
2.1.1 Tivoli SA MP components.....	20
2.1.2 Terminology of Tivoli SA MP.....	22
2.2 How DB2 works with Tivoli SA MP	24
2.2.1 How Tivoli SA MP detects failures	24
2.3 Planning the high availability cluster	27
2.4 Setting up Tivoli SA MP with DB2 10.1 on AIX.....	29
2.4.1 Planning the cluster domain	30
2.4.2 Installing Tivoli SA MP	31

2.4.3 Configuration of Tivoli SA MP and DB2	34
2.5 Administration	56
2.5.1 The node maintenance scenario.....	56
2.6 Cluster maintenance	59
2.6.1 Deleting a domain	61
2.7 Testing.....	62
2.7.1 Operating system failure.....	62
2.7.2 Power failure	65
2.7.3 Network failure	65
2.7.4 DB2 instance failure	68
Chapter 3. DB2 and PowerHA SystemMirror	71
3.1 Overview	72
3.2 How DB2 works with PowerHA	73
3.3 Planning the PowerHA cluster.....	76
3.4 Setting up the PowerHA cluster	76
3.4.1 PowerHA cluster setup planning.....	77
3.4.2 PowerHA configuration	79
3.5 Considerations for db2nodes.cfg file.....	90
3.5.1 Modifying the file entry in the start script.....	91
3.5.2 Running the db2start command with the restart option.....	92
3.5.3 Running the db2gcf command with the -u option	93
3.5.4 Using an alias in the hosts file.....	94
3.6 Tuning tips for quick failover	95
3.6.1 Failover of the resources.....	95
Chapter 4. DB2 with Microsoft Windows Failover Cluster.....	101
4.1 Failover Cluster concepts	102
4.1.1 Failover Cluster overview	102
4.1.2 Windows Failover Cluster definitions	104
4.1.3 Managing Failover Cluster	105
4.2 Minimal steps to cluster a DB2 instance	106
4.3 Creating a server cluster	107
4.3.1 Validating your system	107
4.3.2 Creating a cluster in the domain	115
4.4 Installing DB2	120
4.5 Creating a DB2 instance	121
4.6 Manually configuring a DB2 instance	122
4.6.1 Adding the DB2 resource type	123
4.6.2 Creating cluster resources	125
4.6.3 Migrating the DB2 instance to the cluster environment.....	139
4.6.4 Adding a reference to the instance in the other nodes	141
4.6.5 Configuring security settings.....	142

4.7 Using db2mscs to configure a DB2 instance.....	144
4.8 Testing a cluster	148
4.8.1 Creating a SAMPLE database	149
4.8.2 Verifying the DB2 instance communication settings.....	149
4.8.3 Connecting to the database using Data Studio.....	150
4.8.4 Testing failover	153
4.9 Upgrading your instance	155
Chapter 5. DB2 HADR introduction	157
5.1 HADR overview	158
5.1.1 HADR topology	160
5.1.2 HADR synchronization modes	164
5.2 HADR architecture	165
5.3 Terminology.....	167
Chapter 6. HADR setup	173
6.1 Requirements for setting up HADR.....	174
6.1.1 Requirements	174
6.1.2 Parameters	175
6.2 Setup and configuration	175
6.2.1 Preparing the environment	175
6.2.2 Configuration using the HADR setup wizard.....	178
6.2.3 Command-line setup.....	194
6.2.4 Setting up HADR with multiple standby servers	197
6.2.5 HADR log spooling	201
6.3 Basic operation	201
6.3.1 Starting and shutting down	202
6.3.2 Planned takeover	207
6.3.3 Takeover by force	209
6.4 Troubleshooting.....	212
6.4.1 During setup	213
6.4.2 After setup or during normal execution	214
6.4.3 After an HADR disconnects or server failure occurs.....	215
6.4.4 Considerations while running HADR.....	215
6.4.5 Re-establishing HADR after failure	217
Chapter 7. HADR with clustering software	223
7.1 Overview: Why clustering software is needed.....	224
7.1.1 What is clustering software	224
7.1.2 How HADR works in an environment with clustering software	226
7.1.3 What resources should be taken over.....	227
7.2 db2haicu	228
7.2.1 Prerequisites	228
7.2.2 Usage	229

7.2.3 Considerations	232
7.2.4 Troubleshooting	232
7.3 DB2 HADR with Tivoli SA MP configuration for automatic failover on an AIX system	233
7.3.1 Architecture.....	233
7.3.2 Configuration.....	236
7.3.3 Administration	249
7.3.4 Unplanned outages	255
7.4 DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system	266
7.4.1 Architecture.....	266
7.4.2 Configuration.....	268
7.4.3 Testing	277
7.4.4 Administration	299
7.5 Automating HADR takeover with PowerHA.	306
7.5.1 PowerHA and HADR planning	306
7.5.2 Step-by-step configuration overview.....	310
7.5.3 HADR setup	311
7.5.4 PowerHA configuration	312
7.5.5 Preparing the application server scripts	326
7.5.6 Joint test for HADR and PowerHA	329
Chapter 8. HADR monitoring	337
8.1 Introduction to HADR monitoring.....	338
8.2 The db2pd command	340
8.3 The MON_GET_HADR table function.....	343
8.4 HADR monitoring information	345
Chapter 9. DB2 and system upgrades.....	353
9.1 General steps for upgrades in a HADR environment	354
9.2 DB2 fix pack rolling upgrades	354
9.2.1 Rolling upgrade on Linux	355
9.3 DB2 upgrade	364
9.3.1 DB2 version upgrade on Linux	365
9.4 Rolling operating system and DB2 configuration parameter updates ..	375
9.4.1 Procedure	376
Chapter 10. Automatic client reroute.....	377
10.1 ACR overview	378
10.1.1 ACR with HADR	378
10.1.2 ACR in action	380
10.2 ACR tuning	382
10.3 ACR limitations	385
10.4 ACR configuration examples.....	387

10.4.1	ACR with a non-HADR database	387
10.4.2	ACR with a HADR database	389
10.4.3	ACR with a HADR database and PowerHA	390
10.5	Application programming to handle ACR	392
10.5.1	ACR support for Java applications	392
10.5.2	Implementing ACR on the DataSource interface with JDBC	393
10.5.3	ACR exception handling in Java applications	396
Chapter 11.	HADR configuration parameters and registry variables	399
11.1	DB2 HADR configuration parameters	400
11.1.1	Basic configuration parameters.	400
11.1.2	Automatic client reroute configuration parameters	405
11.2	DB2 HADR registry variables	405
11.3	Considerations	407
11.3.1	DB2 transaction performance	408
11.3.2	How to reduce takeover time	409
11.3.3	Seamless takeover	410
11.3.4	Performance implications of HADR_TIMEOUT	410
11.3.5	Applications with a high logging rate.	410
11.3.6	Network considerations.	411
11.3.7	Network performance tips	413
11.3.8	Avoiding transaction loss in a HADR with HA cluster software	416
11.3.9	Avoiding transaction loss by using the peer window.	422
11.3.10	Index logging.	427
11.3.11	Backup from standby image with FlashCopy	428
11.3.12	Replicating load data.	429
11.3.13	Log archive and HADR	431
11.3.14	Database restore considerations	431
Chapter 12.	Backup and recovery	433
12.1	Single system view backup	434
12.1.1	Using single system view backup	434
12.1.2	Considerations	441
12.2	Backup and restore database with snapshot backup	442
12.3	Recover database command	444
12.3.1	Feature summary	444
12.4	Recovery object management.	445
Chapter 13.	Q replication	447
13.1	Introduction to Q replication	448
13.2	Unidirectional setup.	449
13.2.1	Starting Q capture	484
13.2.2	Start Q Apply.	487

Chapter 14. IBM InfoSphere Change Data Capture	491
14.1 Introduction	492
14.2 Architectural overview	492
14.2.1 InfoSphere CDC architecture	493
14.2.2 Transactional integrity and reliability	496
14.3 InfoSphere CDC topologies.....	497
14.3.1 Unidirectional replication.....	499
14.3.2 Bidirectional replication.....	500
14.3.3 Replication to other destinations.....	501
14.3.4 High availability and disaster recovery with InfoSphere CDC	502
14.4 Features and functionality	503
14.4.1 Transformations	503
14.4.2 Replication modes.....	505
14.4.3 Filtering	507
14.4.4 Conflict detection and resolution.....	508
Chapter 15. Geographically dispersed high availability and disaster recovery solutions	511
15.1 PowerHA over extended distances.....	512
15.2 PowerHA data replication components	515
15.2.1 PowerHA with SAN Volume Controller mirroring	515
15.2.2 PowerHA with Geographical Logical Volume Manager	515
15.2.3 Geographical Logical Volume Manager	516
15.2.4 Synchronous and asynchronous data mirroring	517
15.3 Configuring a stand-alone GLVM	519
15.4 Manual failover	530
15.5 Configuring PowerHA with GLVM	531
Appendix A. PowerHA application server scripts	533
A.1 hadr_primary_takeover.ksh	534
A.2 hadr_primary_stop.ksh	536
A.3 hadr_monitor.ksh	539
Appendix B. IBM Tivoli System Automation for Multiplatforms takeover scripts	541
B.1 env file.....	542
B.2 hadr_start.ksh.....	543
B.3 hadr_stop.ksh	548
B.4 hadr_monitor.ksh	549
B.5 planned_takeover.ksh.....	552
B.6 get_hadr_info.fnc	553
Related publications	555
IBM Redbooks	555

Other publications	555
Online resources	557
Help from IBM	557

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 grant 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 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

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

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

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. 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 IBM's application programming interfaces.

Trademarks

IBM, the IBM logo, and 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®	HACMP™	Redbooks (logo) 
DataStage®	IBM®	System p®
DB2 Connect™	Informix®	System Storage®
DB2 Universal Database™	InfoSphere®	System z®
DB2®	LiveAudit™	SystemMirror®
ECKD™	PowerHA®	Tivoli®
Enterprise Storage Server®	pureScale®	WebSphere®
FlashCopy®	Redbooks®	z/OS®

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.	Linux is a trademark of Linus Torvalds in the United States, other countries, or both.	Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
	Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.	UNIX is a registered trademark of The Open Group in the United States and other countries.

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 using enterprise system resources with broad support for clustering software, such as IBM PowerHA® SystemMirror®, IBM Tivoli® System Automation for Multiplatforms (Tivoli SA MP), and Microsoft Windows Cluster Server.

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

This book explains how to configure Cluster software PowerHA, Tivoli SA MP, 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, and the `db2recovery` command, in detail.

This book is intended for database administrators and information management professionals who want to design, implement, and support a highly available DB2 system.

The team who wrote this book

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

Stanislaw Bartkowski is a Senior Software Engineer at IBM Poland. He joined IBM in 2006. He has been working as a DB2 Solution Migration Consultant for 2 years. He has an extensive experience in software development and database solution development.

Ciaran De Buitlear is a DB2 Solutions Consultant with the Software Group in the IBM Innovation Centre in Dublin, Ireland. Ciaran has a degree in Computer Applications from Dublin City University and has 20 years of industry experience with several relational databases. His main experience is in delivering mission-critical solutions that involve database backed websites, database performance, and high availability.

Adrian Kalicki is a DB2 Solution Migration Consultant at IBM Software Lab in Krakow, Poland. He has over 12 years of experience in the field of Information Management. His areas of expertise cover the full application development lifecycle, including database and application design and performance tuning. He holds a number of certificates for both Oracle and IBM databases.

Michael Loster is a Software Engineer with IBM Research & Development, Böblingen. He is a member of the IBM Information Management Technical Ecosystems Team focusing on the migration of database systems and applications from other database products to DB2. In this role, he is frequently involved in projects that are related to customer database installations and the corresponding application deployments. His areas of expertise include information systems, software engineering, and computer engineering. He holds a Master's degree in Computer Science from the Technical University of Braunschweig.

Marcin Marczewski has been an IBM employee since 2005. He is DB2 Solutions Migration Consultant and Team Leader of the Information Management Technology Ecosystem team in CEE, which is a part of IBM Krakow Software Laboratory in Poland. He focuses on various conversion projects to DB2 database by supporting IBM customers and IBM Business Partners in such projects (porting assessments and porting assistance). As part of his job, he also delivers various DB2 training. He is an IBM certified DB2 for Linux, UNIX, and Windows Advanced Database Administrator.

Anas Mosaad is an IT specialist at the IBM Egypt technology development center. He has over 6 years of experience in the software development industry. His expertise includes portal and J2EE development, web content management, and database application development. He is assigned to the IBM Information Management Technology Ecosystem working on database migrations and database tools development.

Jan Nelken is a DB2 Solution Migration Consultant working in IBM Software Lab in Krakow, Poland. Before his current position, he worked at IBM Toronto Lab in Canada in DB2 for Linux, UNIX, and Windows support. He has 40 years of experience in data processing. His areas of expertise include relational and non-relational databases and DB2 problem determination.

Mohamed Soliman is an IT specialist at IBM Egypt Technology Development Center and is a certified DB2 database administrator. Mohamed is a member of IBM Information Management Technology Ecosystem (IMTE). During his five years with IBM, he has worked on infrastructure technology systems, application development, and database administration.

Klaus Subtil is an IT Specialist within the IBM Information Management Technical Ecosystem team. He has more than 25 years experience in the IT industry focusing on relational database technology and application development. Klaus is an IBM Certified Solution Expert on DB2 10.1 Database administration and holds a Master's degree in Business Administration from the Open University Business School, Milton Keynes, UK. In his current position, Klaus enables independent software vendors, clients, and system integrators for IBM Information Management software.

Marko Vrhovnik is a DB2 migration consultant in the IBM Information Management Technology Ecosystem (IMTE) organization working in Germany. He has over two years of experience in the field of DB2. He works closely with IBM Business Partners and clients regarding DB2, from strategy to implementation, including database migrations and DB2 skill transfer. Marko worked for 5 years as a database researcher in the area of SQL optimization in data-intensive workflows. He holds both a PhD and M.Sc. in Computer Science from the University of Stuttgart, Germany.

Karol Zimnol is a Solution Migration Specialist at IBM, where most of his time is spent on helping IBM customers enable their applications on DB2 and IBM Informix®. The rest of his time is spent on delivering training for IBM Business Partners. He has experience with helping IBM Business Partners to implement high availability applications. Karol has years of experience with DBMS and is certified for DB2, Informix, and Oracle.

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

Lui Tang
IBM Canada

Slawomir Ksiazek and Daniel Tarasek
IBM Poland

Eugene Melody, Mark Dennehy, Sasa Kajic
IBM Ireland

Whei-Jen Chen
International Technical Support Organization, San Jose Center

Thanks to the authors of the previous editions of this book.

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

Whei-Jen Chen
Chandramouli Chandrasekaran
Don Gneiting
Gustavo Castro
Paul Descovich
Tomohiro Iwahashi

- ▶ Authors of the second edition, *High Availability and Disaster Recovery Options for DB2 for Linux, UNIX, and Windows*, published in February 8th, 2009, were:

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

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

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 publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>

Summary of changes

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

Summary of Changes
for SG24-7363-02
for *High Availability and Disaster Recovery Options for DB2 for Linux, UNIX, and Windows*
as created or updated on October 18, 2012.

October 2012, Third Edition

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

New information

- ▶ The DB2 HADR multiple-standbys feature is described.
- ▶ New DB2 HADR parameters are described.
- ▶ DB2 10.1 HADR enhancements are covered.
- ▶ A new chapter that describes Change Data Capture is added.
- ▶ Geographically dispersed options are added.

Changed information

- ▶ All examples are adjusted to DB2 10.1 for Linux, UNIX, and Windows, IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP), and PowerHA.
- ▶ All Control Center examples are replaced with Data Studio 3.1.1.
- ▶ The HADR monitoring methods description is updated to DB2 10.1.
- ▶ Out-of-date examples are dropped.



DB2 high availability and disaster recovery overview

DB2 supports a number of software and hardware offerings from IBM and other vendors that you can use with DB2 to strengthen high availability in your environment. The decision about what products or DB2 features to use depends on the specific challenges of the environment, budget, complexity, and time to implement.

This book describes multiple options for implementing high availability and disaster recovery solutions with DB2. It covers topics such as the integration of various kinds of clustering software with DB2 and the DB2 High Availability and Disaster Recovery (HADR) Feature, and different backup and recovery strategies, including Q replication and IBM InfoSphere® Change Data Capture (CDC).

This chapter provides an overview of those different high availability and disaster recovery options, where it briefly describes some features and products that can help you increase the availability of your applications that are running on DB2.

This chapter covers the following topics:

- ▶ Introduction
- ▶ High availability solutions with DB2
- ▶ Disaster recovery solutions with DB2

1.1 Introduction

This section introduces and explains *high availability* and *disaster recovery* terminology.

1.1.1 High availability

The availability of a database solution is a measure of how successful user applications are at performing their required database tasks. If user applications cannot connect to the database, or if their transactions fail because of errors or time out because of load on the system, the database solution is not available.

It does not matter to a user why the database request failed. Whether a transaction timed out because of bad performance, a component of the solution failed, or an administrator took the database offline to perform maintenance, the result is the same to the user: the database is unavailable to process requests.

Unexpected system failures that could affect the availability of your database solution to users include power interruption, network outage, hardware failure, operating system or other software errors, and complete system failure in the event of a disaster. If such a failure occurs at a time when users expect to be able to do work with the database, a highly available database solution must:

- ▶ Shield user applications from the failure without appreciable performance degradation (or even loss of availability), so the user applications are not aware of the failure.
- ▶ Respond to the failure to contain its effect. For example, if a failure occurs on one machine in a cluster, the cluster manager can remove that machine from the cluster so that no further transactions are routed to be processed on the failed machine.
- ▶ Recover from the failure to return the system to normal operations. For example, if a standby database takes over database operations for a failed primary database, the failed database might restart, recover, and take over once again as the primary database.

These three tasks must be accomplished with a minimal effect on the availability of the solution to user applications.

In addition, in a highly available database solution, the impact of maintenance activities on the availability of the database to user applications must be minimized as well.

For example, if the database solution serves a traditional store front that is open for business between the hours of 9 a.m. - 5 p.m., then maintenance activities can occur offline, outside of those business hours without affecting the availability of the database for user applications. If the database solution serves an online banking business that is expected to be available for customers to access through the Internet 24 hours per day, then maintenance activities must be run online, or scheduled for off-peak activity periods to have minimal impact on the availability of the database to the customers.

When you are making business decisions and design choices about the availability of your database solution, you must weigh the following two factors:

- ▶ The cost to your business of the database being unavailable to customers.
- ▶ The cost of implementing a certain degree of availability.

For example, consider an Internet-based business that makes a certain amount of revenue every hour the database solution is serving customers. A high availability strategy that saves 10 hours of downtime per year earn the business 10 times extra revenue per year. If the cost of implementing this high availability strategy is less than the expected extra revenue, it can be worth implementing.

High availability versus continuous availability

Availability is about ensuring that a database system or other critical server functions remain operational both during planned or unplanned outages, such as maintenance operations, or hardware or network failures. If there is no downtime during planned or unplanned outages, a system is said to be *continuously available*; otherwise, it is said to be *highly available*. With DB2 10.1, it is possible to implement continuously and highly available database applications.

You can use the IBM DB2 pureScale® Feature to implement a continuously available database application. You can use the DB2 pureScale Feature to scale a database across a set of servers in an *active/active* approach: Traffic that is intended for a failed node is either passed on to an existing node or load balanced across the remaining nodes. This technology is similar to the proven data-sharing architecture found in DB2 for IBM z/OS®. For the first time, the benefits of transparency, continuous availability, and scalability are available at a much lower operational cost than ever before. The DB2 pureScale system is run on multiple hosts that access shared data simultaneously, without the need to explicitly modify the application. You can use this transparency to perform maintenance operations on hosts, add more hosts, and remove unnecessary hosts, without impacting your application.

This book describes how to implement a highly available database application with DB2.

For more information about the DB2 pureScale Feature, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.qb.server.doc%2Fdoc%2Fc0056035.html>

Increasing high availability

To increase the availability of a database solution, all of the following issues should be considered:

- ▶ Redundancy: Having secondary copies of each component of your solution that can take over workload in the event of failure. If a component of the system is not redundant, that component could be a single point of failure for the system.
- ▶ System monitoring: Collecting statistics about the components of your solution to facilitate workload balancing or detect that components failed.
- ▶ Load balancing: Transferring some workload from an overloaded component of your solution to another component of your solution that has a lighter load.
- ▶ Failover: Transferring all workload from a failed component of your solution to a secondary component. When workload is transferred this way, the secondary system is said to take over the workload of the failed primary system.
- ▶ Maximizing performance: Reducing the chance that transactions take a long time to complete or time out.
- ▶ Minimizing the impact of maintenance: Scheduling automated maintenance activities and manual maintenance activities to impact user applications as little as possible.
- ▶ Clustering: A cluster is a group of connected machines that work together as a single system. When one machine in a cluster fails, cluster managing software transfers the workload of the failed machine onto other machines.
- ▶ Database logging: Database logging is an important part of your highly available database solution design because database logs make it possible to recover from a failure, and they make it possible to synchronize primary and secondary databases.

Section 1.2, “High availability solutions with DB2” on page 6 introduces important features and products you can use with DB2 10.1 to implement a highly available DB2 application.

For more information about increasing the high availability of database solutions, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0051337.html>

1.1.2 Disaster recovery

The term *disaster recovery* is used to describe the activities that need to be done to restore a database in the event of a fire, earthquake, vandalism, or other catastrophic events that lead to a destruction of the complete database server.

A plan for disaster recovery can include:

- ▶ A remote site to be used in the event of an emergency.
- ▶ A different machine on which to recover the database.
- ▶ Offsite storage of either database backups, table space backups, or both, and archived logs.

In addition, you should answer the following questions to choose a correct disaster recovery solution:

- ▶ How much time can be spent recovering the database (defines the recovery time objective (RTO)).
- ▶ How much data can you afford to lose after recovery (defines 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 be kept?
- ▶ Are table space level backups sufficient, or are full database backups necessary?
- ▶ Is there a need for a hot site for disaster recovery?
- ▶ Should a standby system be configured manually or automatically by using, for example, HADR?

DB2 provides several options when you plan disaster recovery. Based on your business needs, you might decide to use table space or full database backups as a safeguard against data loss, or you might decide that your environment is better suited to a solution like HADR. Section 1.3, “Disaster recovery solutions with DB2” on page 11 introduces functionality that you can use with DB2 10.1 to implement a disaster recovery solution.

For more information about disaster recovery, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.lu.admin.ha.doc%2Fdoc%2Fc0005945.html>

1.2 High availability solutions with DB2

This section introduces important functionality that you can use with DB2 10.1 to implement highly available DB2 applications.

1.2.1 High Availability Disaster Recovery (HADR)

The *High Availability Disaster Recovery (HADR)* feature provides a high availability solution for both partial and complete site failures. In a HADR environment, log data is shipped continuously from a primary database to one or more standby databases and reapplied to the standby databases. When the primary database fails, applications are redirected to a standby database that automatically takes over the role of the primary database.

A partial site failure can be caused by a hardware, network, or software (DB2 or operating system) failure. Without HADR, a partial site failure requires restarting the database management system server that contains the database. The length of time that it takes to restart the database and the server where it is located is unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. With HADR, a standby database can take over in seconds. Further, you can redirect the clients that used the original primary database to the new primary database by using *DB2 automatic client reroute* (see Chapter 10, “Automatic client reroute” on page 377) or try logic again in the application.

This book introduces the DB2 HADR feature in detail, covering setup, administration, and monitoring of an HADR environment and HADR preferred practices. In addition, it describes how to combine HADR with clustering software to increase the availability of a database application that is running on DB2.

For more information about HADR, see the Information Center at <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0011267.html> and *Data Recovery and High Availability Guide and Reference*, SC27-3870-00.

1.2.2 DB2 high availability (HA) feature

The DB2 high availability (HA) feature enables integration between DB2 and cluster managing software.

The idea of clustering is to present to the users a single machine, when in fact the system has multiple nodes to serve client applications. Many clusters act in an active/passive configuration where only one node performs work, with the other nodes that are standing by as the backup if there is a 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.

When you stop a database manager instance in a clustered environment, you must make your cluster manager aware that the instance is stopped. The DB2 High Availability 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, which require changes to the cluster.

The DB2 High Availability Feature is composed of the following elements:

- ▶ IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) is bundled with DB2 10.1 on IBM AIX® and Linux as part of the DB2 HA Feature, and integrated with the DB2 installer. On Windows operating systems, Tivoli SA MP is bundled with DB2 10.1 as well, but it is not integrated with the DB2 database product 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 that you can use with DB2 10.1 can be broadly categorized into two types:

- ▶ Operating system dependent
- ▶ Operating system independent

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.

If there is 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.

Regardless of cluster implementation, database applications require embedded logic to detect lost connections and to try again 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 situation reduces downtime and human intervention.

DB2 10.1 supports the following operating system-dependent cluster managing software:

- ▶ IBM PowerHA SystemMirror for AIX (formerly known as High Availability Cluster Multi-Processing for AIX or IBM HACMP™)
- ▶ Microsoft Cluster Server for Windows
- ▶ Multi-Computer/ServiceGuard, for Hewlett-Packard
- ▶ Sun Cluster for Solaris

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

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.

Chapter 3, “DB2 and PowerHA SystemMirror” on page 71 explains how to integrate PowerHA SystemMirror with DB2 on AIX. Chapter 4, “DB2 with Microsoft Windows Failover Cluster” on page 101 describes the procedure to implement DB2 cluster with Microsoft Windows.

For more information about operating system-dependent clustering solutions, see:

- ▶ IBM PowerHA SystemMirror for AIX:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.lu.admin.ha.doc%2Fdoc%2Fc0007500.html>

- ▶ Microsoft Clustering support for Windows:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0007402.html>
- ▶ Sun Cluster for Solaris:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0007302.html>

Operating system independent solutions

Operating system independent 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 solutions allow frameworks that manage and interchange resources among groups of clusters.

Because these solutions are highly specialized, they offer typically more functionality than their operating system-dependent counterparts. DB2 10.1 supports the following operating system independent cluster managing software:

- ▶ Tivoli SA MP
- ▶ Veritas Cluster Server

Operating system independent clustering software are platform independent, and they offer a single interface across different platforms.

You can use operating system independent clustering software:

- ▶ When you have a heterogeneous environment with different hardware providers.
- ▶ If platform change is a possibility.
- ▶ If your operating system does not offer a cluster solution or has restrictions that can be avoided with an operating system independent clustering software.
- ▶ Suppose that you must centrally manage a complex set of existing base-level clusters on differing platforms, which individually comprise only part of a larger overall application. In this case, a multitiered cluster management solution, such as Tivoli SA MP, can plug in to existing base-level clusters without any requirement to swap architecture or vendors.

Tivoli SA MP components are bundled with DB2 10.1 on Linux and AIX. Chapter 2, “DB2 with IBM Tivoli System Automation for Multiplatforms” on page 19 explains how to implement a DB2 cluster with Tivoli SA MP. Chapter 7, “HADR with clustering software” on page 223 has details about using DB2 10.1 HA Feature to use clustering software on DB2 with HADR.

For more information about operating system independent clustering solutions, see:

- ▶ IBM Tivoli System Automation for Multiplatforms (Linux and AIX):
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0024553.html>
- ▶ Veritas Cluster Server:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0007308.html>

1.2.3 High availability through disk mirroring

Mirroring is the process of writing data to two separate hard disks at the same time. One copy of the data is called a mirror of the other mirror. You can use disk mirroring to maintain a secondary copy of your primary database.

You can use DB2 10.1 suspended I/O functionality to split the primary and secondary mirrored copies of the database without taking the database offline. After the primary and secondary databases copies are split, the secondary database can take over operations if the primary database fails.

For more information about disk mirroring, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0006356.html>

1.2.4 High availability through log shipping

Log shipping is the process of copying log files from a primary to a secondary database copy either from an archive device, or through a user exit program that is running against the primary database.

When the production machine fails, a failover occurs and the following actions must take place:

- ▶ The remaining logs are transferred over to the standby machine.
- ▶ The standby database rolls forward to the end of the logs.
- ▶ The clients reconnect to the standby database and resume operations.

When you use this approach, the primary database is restored to the standby machine by using the DB2 10.1 restore utility or the split mirror function.

To ensure that you are able to recover your database in a disaster recovery situation, consider the following items:

- ▶ The archive location should be geographically separate from the primary site.
- ▶ Remotely mirror the log at the standby database site. Mirroring log files helps protect a database from accidental deletion of an active log and data corruption that is caused by hardware failure.

For more information about log shipping and mirroring, see:

- ▶ Log shipping:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0007190.html>

- ▶ Log mirroring:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0006077.html>

1.2.5 Automatic client reroute

Automatic client reroute (ACR) is a DB2 10.1 feature that you can use in a high available database solution to redirect client applications from a failed server to an alternate server so the applications can continue their work with minimal interruption. ACR can be accomplished only if an alternate server is specified before the failure.

Chapter 10, “Automatic client reroute” on page 377 describes ACR in detail.

For more information about automatic client reroute, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fr0023392.html>

1.3 Disaster recovery solutions with DB2

This section introduces important functionality that you can use with DB2 10.1 to implement disaster recovery solutions.

1.3.1 Backup and recovery options

Backup and recovery is the fundamental technology for a disaster recovery 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 have to back up only changed data since your last backup; it is faster than a full backup. DB2 also provides automatic backup functionality; you define your backup policy and DB2 backs up your database automatically.

In a partitioned database environment, DB2 supports *Single System View (SSV)* backups, which you can use to back up a database on all database partitions or a subset of database partitions with one **backup** command. All backup images have the same timestamps and include the necessary logs to roll forward to a consistent point. Section 12.1, “Single system view backup” on page 434 provides details about SSV backups.

DB2 *Advanced Copy Service (ACS)* supports snapshot functionality, which is based on the IBM FlashCopy® technology of the IBM Tivoli Storage Manager solution. When you use this functionality, you can make nearly instantaneous point in time copies of entire logical volumes or data sets with minimal impact on your production system. DB2 ACS is bundled with DB2 starting in Version 10.1. Chapter 12, “Backup and recovery” on page 433 provides an overview of snapshot backups and restores.

The recreation of the database is called *recovery*. Version recovery is the restoration of a previous version of the database, using an image that was created during a backup operation. Rollforward recovery is the reapplication of transactions that are recorded in the database log files after a database or a table space backup image is restored. Both recovery types are fundamental for a disaster recovery solution.

If your plan for disaster recovery is to restore the entire database on another machine, have at least one full database backup and all the archived logs for the database. Although it is possible to rebuild a database if you have a full table space backup of each table space in the database, this method might involve numerous backup images and be more time-consuming than recovery using a full database backup.

You can choose to keep a standby database up to date by applying the logs to it as they are archived. Or, you can choose to keep the database or table space backups and log archives in the standby site, and perform restore and rollforward operations only after a disaster occurs. (In the latter case, recent backup images are preferable.) In a disaster situation, however, it is generally not possible to recover all of the transactions up to the time of the disaster.

The usefulness of a table space backup for disaster recovery depends on the scope of the failure. Typically, disaster recovery is less complicated and time-consuming if you restore the entire database; therefore, a full database backup should be kept at a standby site. If the disaster is a damaged disk, a table space backup of each table space on that disk can be used to recover. If you lose access to a container because of a disk failure (or for any other reason), you can restore the container to a different location.

For more information about how to develop a backup and recovery strategy, see:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0005945.html>

1.3.2 High Availability Disaster Recovery (HADR)

Another way that you can protect your data from complete site failures is to implement the DB2 high availability disaster recovery (HADR) feature that is described in 1.2.1, “High Availability Disaster Recovery (HADR)” on page 6. After it is set up, HADR protects against data loss by replicating data changes from a primary database to one or more standby databases. By using HADR, you can improve the performance of a disaster recovery operation in comparison to conventional methods.

A complete site failure can occur when a disaster, such as a fire, causes the entire site to be destroyed. However, because HADR uses TCP/IP for communication between the primary and standby databases, they can be situated in different locations. For example, the primary database might be at your head office in one city, and a standby database might be at your sales office in another city. If a disaster occurs at the primary site, data availability is maintained by having the remote standby database take over as the primary database with full DB2 functionality. After a takeover operation occurs, you can bring the original primary database back up and return it to its primary database status; this action is known as *failback*. You can initiate a failback if you can make the old primary database consistent with the new primary database. After you reintegrate the old primary database into the HADR setup as a standby database, you can switch the roles of the databases to enable the original primary database to once again be the primary database.

This section describes the DB2 HADR Feature in detail, covering setup, administration, and monitoring of an HADR environment, and HADR preferred practices. In addition, it describes how to combine HADR with clustering software to increase the availability of a database application that is running on DB2.

For more information about HADR, see the Information Center at <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0011267.html> and *Data Recovery and High Availability Guide and Reference*, SC27-3870-00.

1.3.3 Replication

You can also protect your data from partial or complete site failures by using replication. You can use replication to copy data regularly to multiple remote databases. DB2 database provides a number of replication tools that you can use to specify what data should be copied, which database tables the data should be copied to, and how often the updates should be copied.

Data replication is the process of capturing data changes from data sources on a source server and reapplying these changes to corresponding data sources on a target server. In this way, you can use data replication as an alternate technology to build a hot standby server, especially in a heterogeneous environment.

For this purpose, IBM InfoSphere Replication Server provides two different data replication solutions that are integrated into DB2 to replicate data from and to relational data sources: *SQL replication* and *Q replication*.

SQL replication

SQL replication is a solution that captures changes to source tables and views and uses staging tables to store data changes of committed transactions. The changes are then read from the staging tables and replicated to the corresponding target tables.

Figure 1-1 illustrates the infrastructure for a simple configuration in SQL replication.

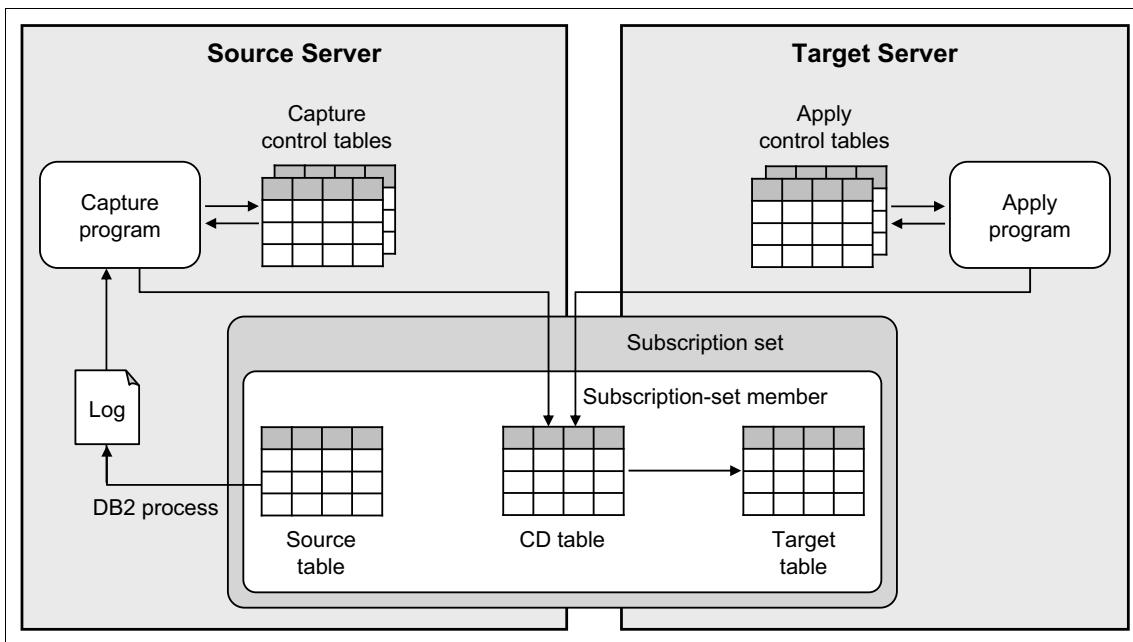


Figure 1-1 Infrastructure for a simple configuration in SQL replication

The *Capture program* that is running on the source server scans the DB2 log files for changes that belong to a source table and then places this information in staging tables that are known as *change-data (CD) tables*. The *Apply program* that is running on the target server reads the CD tables and replicates the data changes to corresponding target tables. Inside the source server are a set of DB2 relational tables that are called *Capture control tables* that contain information about source tables. The Capture program uses this information to know what data it is supposed to capture. Information about target tables goes in to the *Apply control tables*, which are typically on a target server. The Apply program uses this information to know which targets it is supposed to write data to. A *subscription set* defines the mapping between source and target tables.

Q replication

Q replication is a solution that captures changes to source tables and converts data changes of committed transactions to messages. In contrast to SQL replication, the data is not staged in tables. As soon as the data is committed at the source and read by Q replication, the data is sent to the target location through IBM WebSphere® MQ message queues. At the target location, the messages are read from the queues and converted back into transactional data that is applied to the target tables. With Q replication, you can configure three different types of replication: *unidirectional*, *bidirectional*, and *peer-to-peer*.

Figure 1-2 illustrates the infrastructure for a simple unidirectional configuration in Q replication, where changes that occur to a source table on a source server are replicated to a target table on a target server.

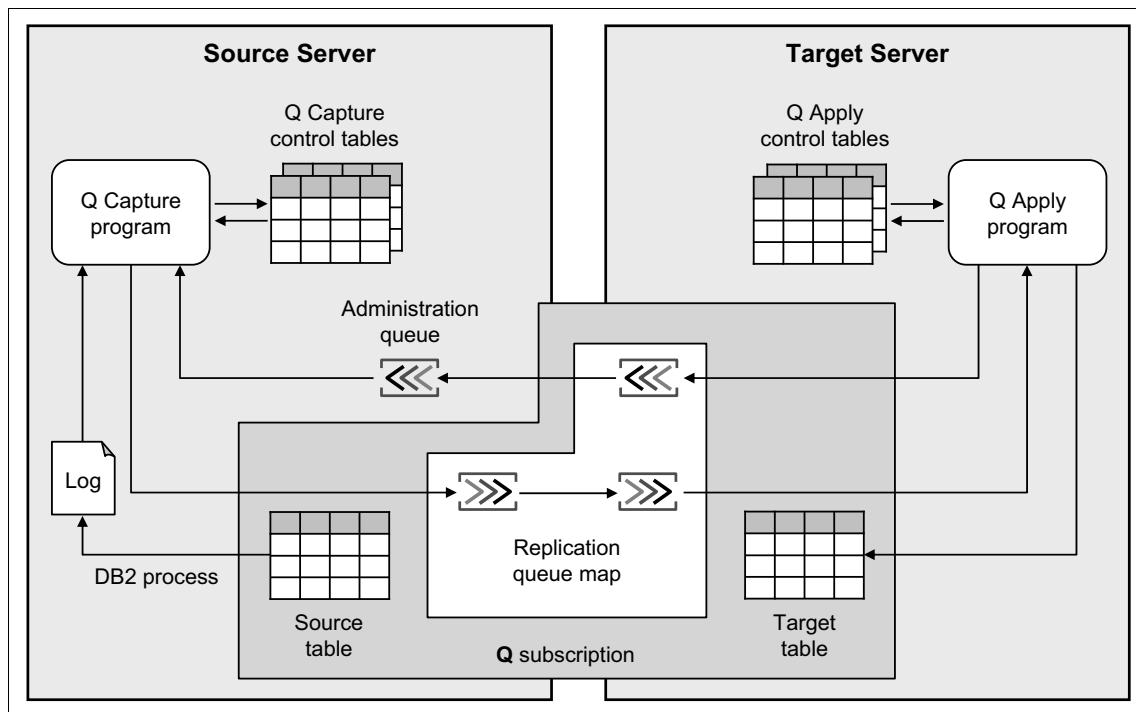


Figure 1-2 Infrastructure for a simple configuration in Q replication

The *Q Capture program* that is running on the source server reads the log files for changed source data and writes the changes to WebSphere MQ queues. The *Q Apply program* that is running on a target server retrieves the captured changes from the queues and writes the changes to a target table. The Q Capture and Apply program use a set of control tables that is called *Q Capture and Apply control tables* that contain information about the replication sources and targets, and which WebSphere MQ queues to use. A *Q subscription* defines the mapping between the source and target tables.

Chapter 13, “Q replication” on page 447 describes in detail how to set up a unidirectional configuration in Q replication.

For more information about replication solutions with DB2 10.1, see:

- ▶ IBM InfoSphere Replication Server:
http://www-01.ibm.com/software/data/infosphere/replication-server/features.html?S_CMP=rnav
- ▶ SQL replication:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.swg.im.iis.db.rep1.sqlrep1.doc%2Ftopics%2Fiiyrsncsqlreplovu.html>
- ▶ Q replication:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.swg.im.iis.db.rep1.intro.doc%2Ftopics%2Fiiyrcintrsqr0.html>

1.3.4 InfoSphere Change Data Capture (CDC)

InfoSphere CDC is a replication solution that captures and delivers database changes as they happen. InfoSphere CDC allows replication between heterogeneous environments, supporting various operating systems and databases. Destinations for InfoSphere CDC replication include databases (which can be different from the source database), message queues, or an ETL solution such as IBM InfoSphere DataStage®.

What gets replicated depends on table mappings that are configured in the InfoSphere CDC. InfoSphere CDC employs a non-invasive approach to capturing changes that take place on the source database, reading changes directly from database logs. No changes are required to the source application. The capture mechanism (log scraping) is a lightweight process that runs on the source server and avoids significant impact on production systems.

Chapter 14, “IBM InfoSphere Change Data Capture” on page 491 describes InfoSphere CDS in detail.

For more information about InfoSphere CDC, see:

<http://publib.boulder.ibm.com/infocenter/cdc/v6r5m1/index.jsp>

1.3.5 Remote disk mirroring

The concept behind remote disk mirroring is to have two or more disk arrays in boxes (also referred to as *storage enclosures*), in geographically dispersed locations. These boxes are connected using *dark fiber* or similar technologies and implement algorithms to replicate the data between the boxes.

Remote disk mirroring replicates all disk writes with minimal delay, 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. 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 IBM 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)

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



DB2 with IBM Tivoli System Automation for Multiplatforms

This chapter explains how to integrate DB2 in an IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) environment. It provides the basic management concepts with considerations to reduce the time that is consumed for failover.

This chapter covers the following topics:

- ▶ Overview
- ▶ How DB2 works with Tivoli SA MP
- ▶ Planning the high availability cluster
- ▶ Setting up Tivoli SA MP with DB2 10.1 on AIX
- ▶ Administration
- ▶ Cluster maintenance
- ▶ Testing

2.1 Overview

Tivoli SA MP provides a framework to manage automatically the availability of components that are 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 Tivoli SA MP is granted access. That is, Tivoli SA MP manages the availability of any IP address that a user wants by floating that IP address among NICs to which it is access.

For example, both a DB2 instance and the file systems that are used by DB2 have start, stop, and monitor commands. Therefore, Tivoli SA MP scripts can be written to manage these resources automatically. Scripts, and other attributes of a resource, are required by Tivoli SA MP to manage that resource. Tivoli SA MP stores a resource's attributes in an object container, much like the attributes of a Java class. In fact, Tivoli SA MP manages a resource by instantiating a class for that resource.

You can use Tivoli SA MP to manage related resources in resource groups. If you use Tivoli SA MP, you can ensure that all resources within a resource group are online at only one physical node at any point in time. Also, all of those resources are 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, Tivoli SA MP 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.

2.1.1 Tivoli SA MP components

Since V2.1, Tivoli SA MP consists of these components:

- ▶ *End-to-end automation component*

This is the second-tier-enabling technology, which gives Tivoli SA MP a rare ability to provide control over heterogeneous environments of multiple base-level clusters. The low-level clustering software that is installed is not replaced.

Tivoli SA MP end-to-end automation can control an existing PowerHA on AIX, Heartbeat on Linux, or MSCS as though it were controlling a Tivoli SA MP base component cluster on these same platforms. Tivoli SA MP software interfaces that are called *End-to-End Automation Adapters* operate between Tivoli SA MP and the base-level clustering software.

► *Base component*

This Tivoli SA MP component provides base-level cluster functionality. It acts as an equivalent to PowerHA or other clustering software, on AIX and Linux. The Tivoli SA MP base component consists of the following parts:

– *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 that exists within a single base-level cluster, and the end-to-end automation layer that spans multiple clusters.

– *Operations console of the base component*

The Tivoli SA MP Operations Console is a web browser accessed feature (no client software installation is required), based on the WebSphere Portal Server Integrated Solutions Console. Because it spans all Tivoli SA MP domains, and provides the same interface, there is no interface discrepancy between platforms.

Although 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:

- *End-to-end automation mode*: Where end-to-end automation is active.
- *First-level automation mode*: Where end-to-end automation is installed, but is not active.
- *Direct-access mode*: Where you are essentially running the console directly on a system that is part of a base-level cluster. Tivoli SA MP 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-Tivoli SA MP clustering software.

For more information, see *Tivoli System Automation for Multiplatforms Version 3.2.2 Administrator's and User's Guide*, SC34-2583-03.

2.1.2 Terminology of Tivoli SA MP

The following terms are common between Tivoli SA MP 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, middle ware and DBMS, or WebSphere Administration Server and web server daemons. Regarding up/down status monitoring and automation (and end-to-end automation), resources can also exist at multiple tiers, for example, grouping granular resources and base-level cluster environments to create 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 is a static IP address that is shared between the nodes that 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. Resources are part of these domains, so if groups of resources become unavailable, a node or subcluster is unavailable. This situation 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 level of operational nodes that are required before certain operations can be performed within a cluster. Tivoli SA MP 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. There are six types of the tiebreaker, depending on the platform and devices available:
 - *Operator*: Requires human intervention for a decision 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. The **SCSI reserve** or **persistent reserve** command is used to achieve tie-breaker exclusive status.
 - *IBM ECKD™*: Shared ECKD disk on Linux on IBM System z®. The ECKD **reserve** command is used to achieve tie-breaker exclusive status.

- *Disk*: A 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 that fail. This method relies on the failure of an ICMP response to indicate subcluster/node failure and is unable to determine if communication is still working between the subcluster peer nodes.

In Tivoli SA MP specific terminology, the hierarchy in which Tivoli System Automation coordinates resources in a clustered environment consists of an *Automation Manager*, which consists of a *Binder* and a *Logic Deck*.

- ▶ *Binder*: Has resources that are 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 started only if resource dependencies are met.
- ▶ If Tivoli SA MP has not tried to access a 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 a *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 rules-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 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. Tivoli SA MP can choose any of these resources to provide a function in the required definition of availability. For example, if you have a number of redundant network adapters, if one fails, that IP address can be mapped by Tivoli SA MP to another adapter in the equivalency.

Understanding the interaction between the components within this hierarchy is beyond the scope of this book. For more information, see:

- ▶ Tivoli System Automation for Multiplatforms Version 3.2.2 Administrator's and User's Guide, SC34-2583-03.
- ▶ *Tivoli System Automation for Multiplatforms Version 3.2.2 Installation and Configuration Guide*, SC34-2584-03
- ▶ *End-to-end Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117.
- ▶ Getting Started Guide on IBM developerWorks:
<http://www.ibm.com/developerworks/tivoli/library/tv-tivoli-system-automation/index.html>
- ▶ Tivoli System Automation manuals:
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms3.1.html>
- ▶ Tivoli SA MP introduction:
<http://www.ibm.com/software/tivoli/products/sys-auto-multi/index.html>

2.2 How DB2 works with Tivoli SA MP

Technically, Tivoli SA MP 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 situation enables attribution of priorities so that for any given partial or total resource failure scenario, the appropriate action is taken.

2.2.1 How Tivoli SA MP detects failures

Tivoli SA MP controls the DB2 instance and DB2 data objects resources as application resources. For the application resources, Tivoli SA MP detects the failure by using monitor scripts. When you create the application resource, you must register the monitor script to the application resource with a valid argument to detect the failure with a correct return code.

DB2 HA Feature and Tivoli SA MP: If you use the DB2 High Availability (HA) Feature with Tivoli SA MP as your cluster manager, the database manager uses scripts to support automated failover solutions. These scripts are installed or updated automatically when you use the DB2 installer to install or update Tivoli SA MP. When you install or update Tivoli SA MP using the `installSAM` utility, you must then manually install or update these scripts.

The monitor script is run by Tivoli SA MP regularly and returns the return code (RC) to Tivoli SA MP. Tivoli SA MP decides on the next action that is based on this return code. Table 2-1 and Table 2-2 on page 26 list the meanings of the return codes and the actions that Tivoli SA MP takes when the nominal state is online. If RC=1, this resource is recognized to be “Online,” and Tivoli SA MP takes no action. If the monitor script returns a RC=2, this resource is considered to be “Offline,” and Tivoli SA MP first stops the related resources, then start this resource again.

Table 2-1 Return codes from the monitor script and actions that are taken by Tivoli SA MP

RC	OpState	Meaning	First action of Tivoli SA MP (nominal state=online)	Second action of Tivoli SA MP (nominal state=online)
0	Unknown ^a	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. ^b	Nothing.
3	Failed Offline	Failed (cannot recover).	Stop the related resources.	Perform takeover. ^c
4	Stuck Online	The <code>start</code> or <code>stop</code> command times out. User intervention is needed.	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 Tivoli SA MP when the monitor script timeout occurs.
- b. If there are any resources that are related to the failed resource, Tivoli SA MP stops the related resources in advance.
- c. The Mandatory attribute value of this resource should be true.

Table 2-2 Return code from the start/stop script and Tivoli SA MP actions

RC	Meaning	First action of Tivoli SA MP (nominal state=online)	Second action of Tivoli SA MP (nominal state=online)
0	The start or stop command completed successfully.	Move to the next start/stop sequence.	Nothing.
Not 0	The start command failed (Failed Offline).	Stop the resource.	Perform takeover.
Not 0	The stop command failed.	Nothing.	Nothing.

Figure 2-1 illustrates the typical behavior when a resource in one node failed. The return codes 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). Tivoli SA MP runs 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). Tivoli SA MP runs the start script. However, if the start script cannot start the resource, it returns RC=1, which triggers the failover action. Tivoli SA MP then tries to stop the resource on node1. After the stop, Tivoli SA MP fails the resource groups on node1 over to node2.

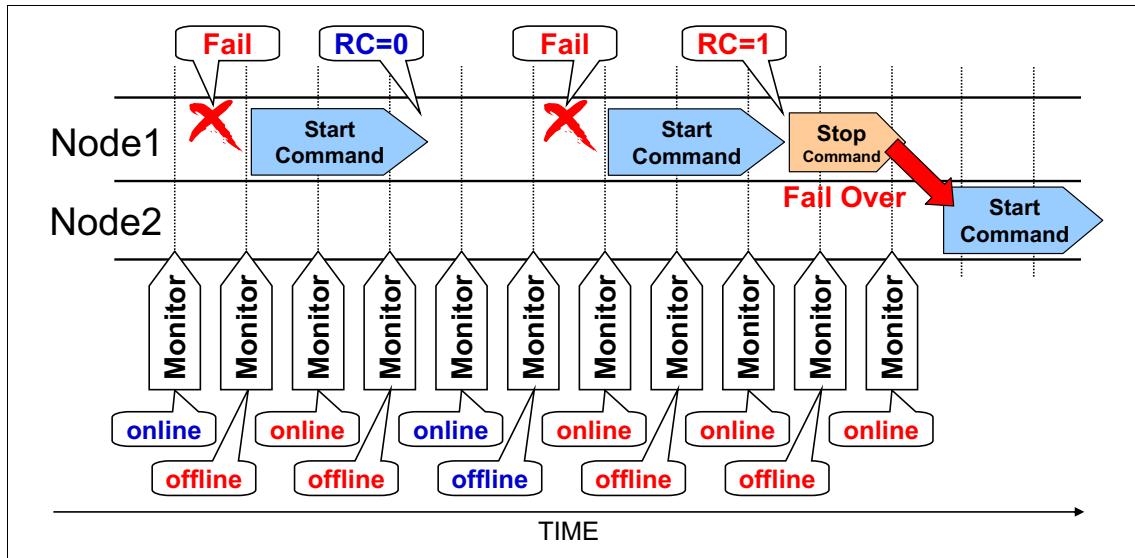


Figure 2-1 The failed resource and recovery

This is a typical recovery behavior. The DB2 resource, the mount point resources, and LVM resource all operate this way.

2.3 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 simple and easy to manage, but one node is not used 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 2-2 illustrates the mutual takeover configuration.

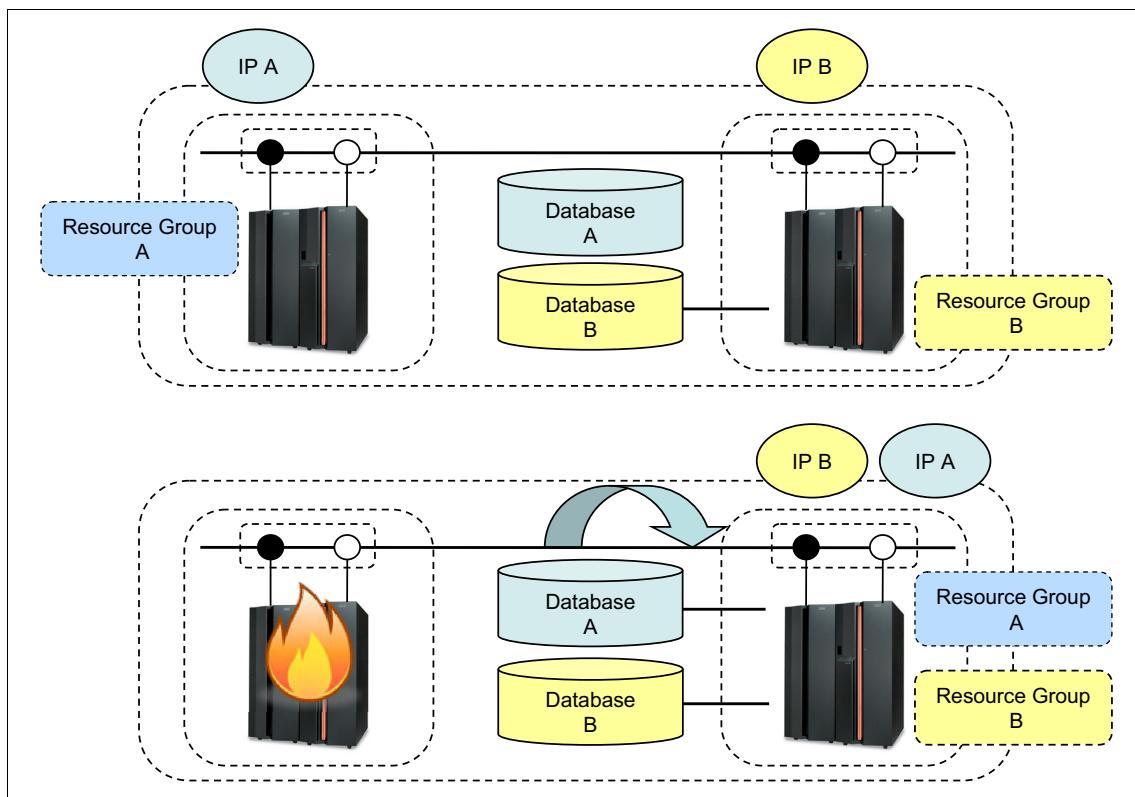


Figure 2-2 Mutual takeover

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, that is, node_B. If node_B crashes, the same operation happens, just in the opposite direction.

This configuration is called active/active because each node has its own database that is 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, which is why this configuration is also called a mutual takeover configuration. With this configuration, you can use both machine resources for services, but this setup is more complicated compared to active/standby.

For this configuration, you must plan carefully so that each node has enough machine resources (processor or physical memory) to carry two databases while it is running. For example, you have 1 GB physical memory on each node and 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 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 (Figure 2-3).

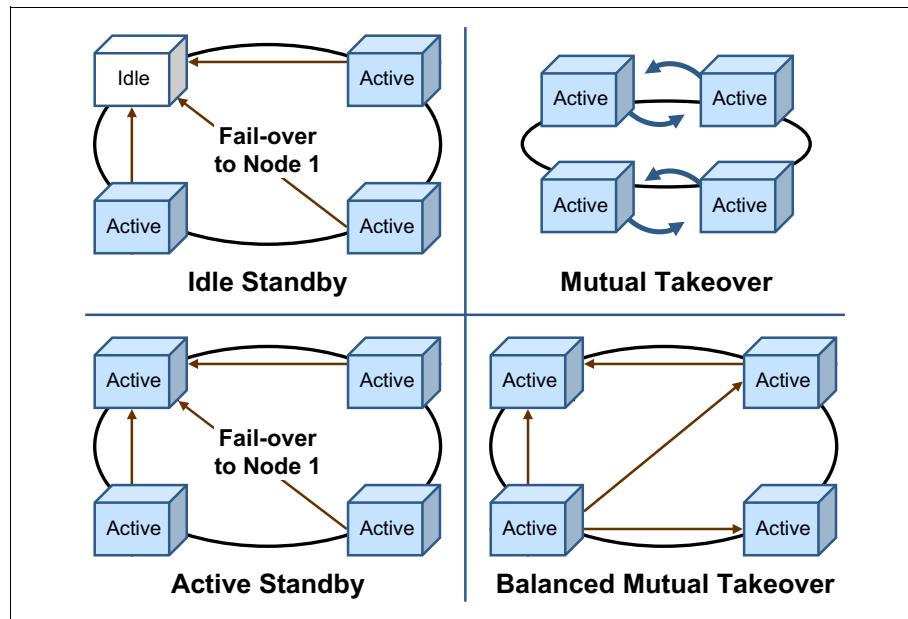


Figure 2-3 Variations of cluster configuration for multiple nodes

For more details about the DB2 highly available data store, see “IBM DB2 Universal Database™ and the Highly Available Data Store” at:

<http://www.ibm.com/developerworks/db2/library/techarticle/0310melnyk/0310melnyk.html>

2.4 Setting up Tivoli SA MP with DB2 10.1 on AIX

This section focuses on the setup procedure for one active/standby cluster with a shared disk. It starts from the planning of the cluster domain, then proceed to the installation, configuration, and, at the end, administration and testing.

Two main steps for setting up the system environment of a cluster domain are:

- Planning the cluster domain
- Configuration of Tivoli SA MP and DB2

2.4.1 Planning the cluster domain

Use the following resources to create a sample cluster domain:

- ▶ Two servers, node1 and node2, running AIX V7.1 TL0 SP03-1115.
- ▶ An external storage that can be accessed from both servers.
- ▶ A public network that both servers can communicate through.
- ▶ Each server can reach the third machine as a tiebreaker.
- ▶ DB2 10.1 with no fix packs with Tivoli SA MP installed on both servers.

Figure 2-4 shows the architecture of this cluster domain. This cluster is a simple one on two nodes. During normal operation, one server in the cluster domain owns all the resources to run the DB2 instance. This server is 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 takes over the resources is called the *failover node*.

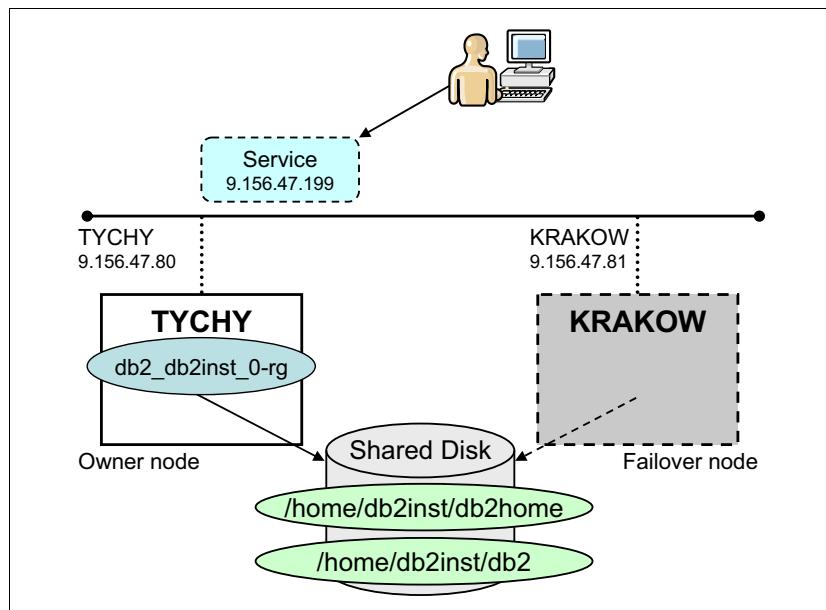


Figure 2-4 The architecture of the sample cluster domain

After the node's role in the cluster domain is decided, you can plan the Tivoli SA MP configuration. The SA MP configuration includes resource groups, resources, equivalencies, relationships, and scripts to control the application resources.

Configure the db2_db2inst2_0-rg resource group and define the resources that are listed in Table 2-3 as the members of this resource group.

Table 2-3 Resources

Resource	Description
db2_db2inst_0-rs	DB2 instance
db2mnt-home_db2inst_db2home-rs	DB2 instance directory mount point in shared disk
db2mnt-home_db2inst_db2-rs	DB2 data directory mount point in shared disk
db2ip_9_156_47_199-rs	Service IP address

In our cluster domain, when Tivoli SA MP starts the DB2 instance, the mount points and LVM and the service IP address must be started.

2.4.2 Installing Tivoli SA MP

Tivoli SA MP is integrated with IBM Data Server as part of the DB2 High Availability Feature on AIX, Linux, and Solaris operating systems. You can install, upgrade, or uninstall Tivoli SA MP using either the DB2 installer or the **installSAM** and **uninstallSAM** scripts that are included in the IBM Data Server installation media. On Windows operating systems, the Tivoli SA MP is bundled as part of the DB2 High Availability Feature, but it is not integrated with the DB2 installer.

DB2 installer

There are three methods for using the DB2 installer to install or uninstall:

1. DB2 Setup wizard (install, upgrade, or uninstall)
2. Silent installation by using a response file with **db2setup** (install or upgrade) or **db2unins** (for uninstall)
3. The **db2_install** command (for installation), **installFixPack** command (for upgrade), or **db2_deinstall** command (for uninstall)

The examples that are described in 2.4.3, “Configuration of Tivoli SA MP and DB2” on page 34 are based on a fresh DB2 10.1 installation on AIX. DB2 is installed with DB2 Setup and two screen captures were taken during the process, as shown in Figure 2-5 and Figure 2-6 on page 33.

For more information, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.qb.server.doc%2Fdoc%2Ft0051289.html>

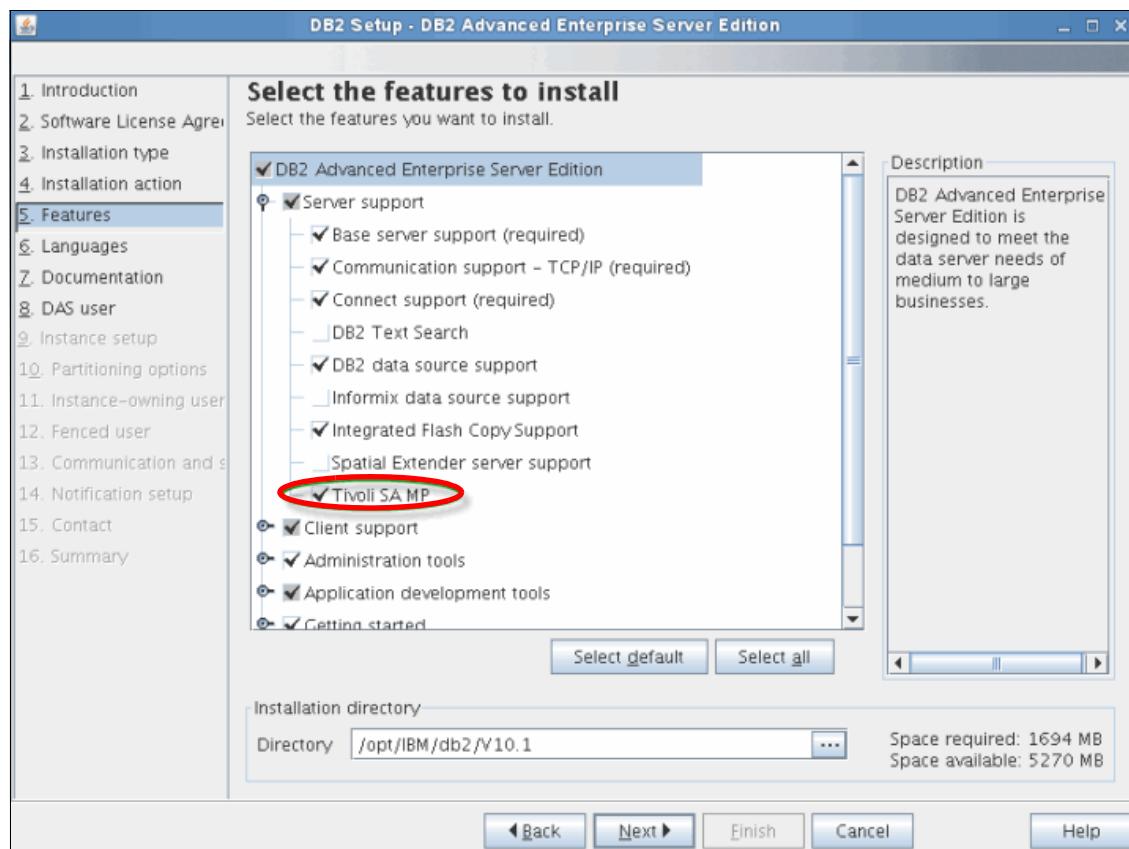


Figure 2-5 DB2 installation - features

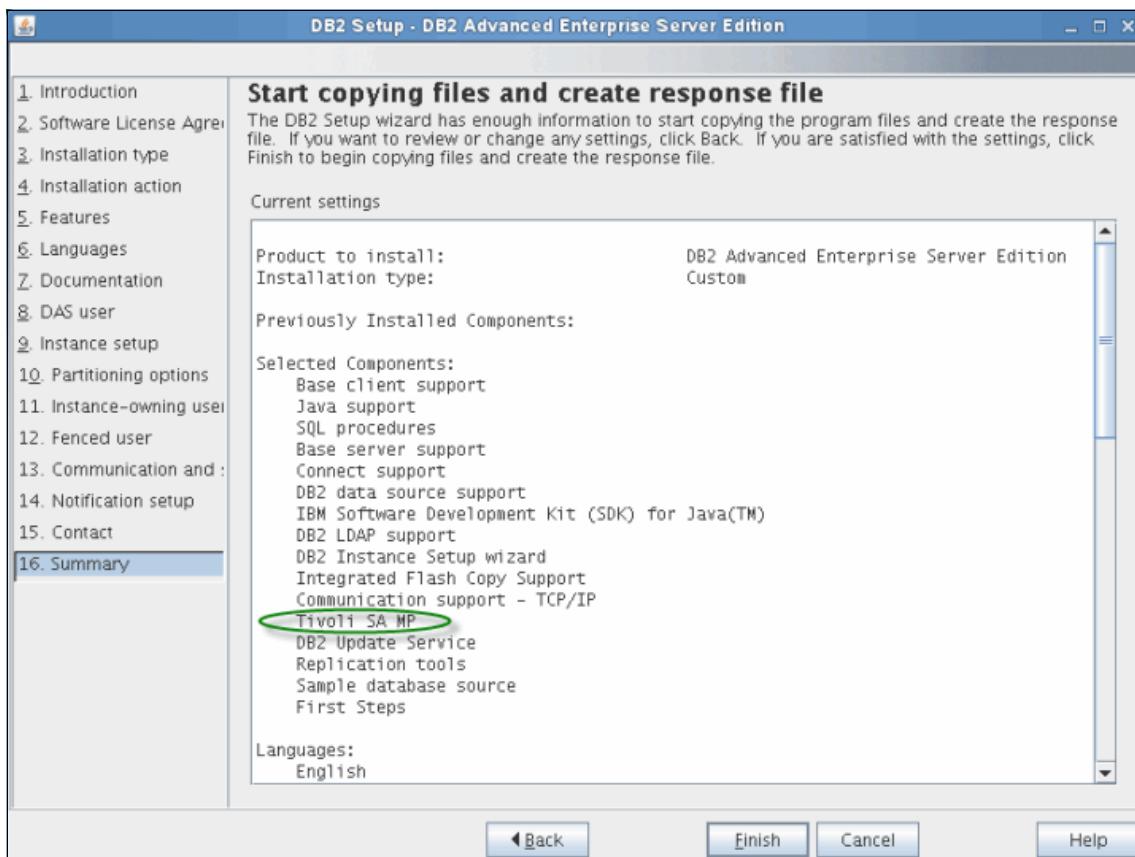


Figure 2-6 DB2 installation - summary

The information about your system that the DB2 installer collects determines which windows open in the graphical interface of the DB2 Setup wizard during installation. For example, if you already have Tivoli SA MP installed, then the DB2 Setup wizard does not display a window to install Tivoli SA MP.

installSAM script

The `installSAM` script is on the IBM DB2 installation media in `db2/platform/tsamp`, where platform refers to the appropriate hardware platform. Example 2-1 shows the `installSAM` location on AIX.

Example 2-1 installSAM location

```
root@TYCHY # cd aese/db2/aix/tsamp  
root@TYCHY # ls
```

AIX	db2cktsa	installSAM	license	prereqSAM
	uninstallSAM			

If you use the DB2 High Availability (HA) Feature with Tivoli SA MP as your cluster manager, the database manager uses scripts to support automated failover solutions. These scripts are installed or updated automatically when you use the DB2 installer to install or update SA MP. When you install or update SA MP using the **installSAM** utility, you must then manually install or update these scripts.

For more information, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luwb.server.doc%2Fdoc%2Ft0051289.html>

2.4.3 Configuration of Tivoli SA MP and DB2

This section explains the configuration procedure for a shared disk HA cluster with DB2 10.1 and Tivoli SA MP. It describes a step-by-step procedure for configuring a cluster domain using the **db2haicu** utility with an XML file. For the setup example using **db2haicu** interactive mode, see 7.3, “DB2 HADR with Tivoli SA MP configuration for automatic failover on an AIX system” on page 233.

The procedure to configure a highly available DB2 environment with shared storage has the following steps:

1. Check prerequisites.
2. Check the network interface.
3. Set up the storage for sharing.
4. Create a DB2 instance.
5. Cluster preparation.
6. System clock synchronization.
7. Creating an XML configuration file for **db2haicu**.
8. Setting up a cluster domain using **db2haicu** with an XML file.
9. Verifying the created cluster domain.

This section describes each step in detail in the following sections.

Checking prerequisites

The first configuration step is checking the prerequisites for **db2haicu**. The details are described in 7.2, “db2haicu” on page 228.

To check the prerequisites, set the environment variable. In our example, we set the environment variable to use the Tivoli SA MP environment. The CT_MANAGEMENT_SCOPE environment variable must be set for all users of Tivoli SA MP on all nodes. Add the following entry to /etc/profile:

```
export CT_MANAGEMENT_SCOPE=2
```

For Linux environment, add the entry to /etc/profile.local.

Verify the environment variable settings (Example 2-2).

Example 2-2 Checking value of environment variables

```
(0)1(F)2 root@TYCHY # cat /etc/profile
..
# add for TSA/DB2 HA Configuration
export CT_MANAGEMENT_SCOPE=2
(0) (F) root@TYCHY # echo $CT_MANAGEMENT_SCOPE
2
```

Checking the network interface.

Assign the following static IP addresses to the en0 adapters on the owner and failover nodes:

- ▶ Owner node (TYCHY):
en0: 9.156.47.80 (255.255.255.0)
- ▶ Failover node (KRAKOW):
en0: 9.156.47.81 (255.255.255.0)

All cluster nodes must have static IP addresses. Ensure that the /etc/hosts file of every cluster node contains entries of the owner node and the failover node names. Example 2-3 shows the entries in the /etc/hosts file of both servers.

Example 2-3 The entries in /etc/hosts on node1 and node2

```
9.156.47.80 TYCHY.kraklab.pl.ibm.com
9.156.47.81 KRAKOW.kraklab.pl.ibm.com
```

¹ (O)- Owner node

² (F)- Failover node

Two servers must be able to communicate with each other through a public network. Confirm this situation by running **ping**. Run the following commands on both nodes and ensure that they complete successfully:

- ▶ (0) (F) \$ **ping** TYCHY
- ▶ (0) (F) \$ **ping** KRAKOW

Now, confirm the broadcast addresses. Tivoli SA MP 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 (Example 2-4).

Example 2-4 Checking the broadcast address

```
(0) root@TYCHY # ifconfig -a
en0:
flags=1e080863,480<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GR
OUPRT,64BIT,CHECKSUM_OFFLOAD(ACTIVE),CHAIN>
        inet 9.156.47.80 netmask 0xffffffff00 broadcast 9.156.47.255
              tcp_sendspace 262144 tcp_recvspace 262144 rfc1323 1

(F) root@KRAKOW # ifconfig -a
en0:
flags=1e080863,480<UP,BROADCAST,NOTRAILERS,RUNNING,SIMPLEX,MULTICAST,GR
OUPRT,64BIT,CHECKSUM_OFFLOAD(ACTIVE),CHAIN>
        inet 9.156.47.81 netmask 0xffffffff00 broadcast 9.156.47.255
              tcp_sendspace 262144 tcp_recvspace 262144 rfc1323 1
```

You can calculate the correct broadcast address from the IP address and the netmask. The broadcast address is computed by the 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 2-4 shows a sample calculation of the broadcast address.

Table 2-4 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 running **smit** and completing the following steps.

1. Start the **smit** menu by running **smitty tcPIP**.
2. Select Further Configuration → Network Interfaces → Network Interface Selection → Change / Show Characteristics of a Network Interface.
3. Select the network interface whose configuration you want to change.

If your platform is Linux, you can use the **ip** command instead.

Setting up the storage for sharing

Before you proceed with the steps to set up the storage for sharing between two nodes, 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 if there is a takeover. Therefore, ensure that both servers can vary on the volume groups.

For the details about storage configuration, see the documents that are provided by your storage vendors, for example, *IBM System Storage DS4000 and Storage Manager V10.30*, SG24-7010.

To complete the shared storage setup, complete the following steps:

1. Ensure that your share storage is accessible from both nodes by running **1spv** (Example 2-5).

Example 2-5 List volume group

(0) (F) root@TYCHY #	1spv	
hdisk0	00cca5e4801553cf	rootvg
active		
hdisk2	none	None

In our example, hdisk2 is the new storage server.

2. Create a volume group on your share disk by running **mkvg** (Example 2-6).

Example 2-6 Create volume group

(0) root@TYCHY #	mkvg -y 'sharevg2'	hdisk2
0516-1254 mkvg: Changing the PVID in the ODM.		
sharevg2		

3. 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 run 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 2-7 shows a command sample and output. In this example, the output shows that the volume group major number of “sharevg” is 35. Repeat these steps for all the volume groups.

Example 2-7 Sample output of special device file

```
(0) root@TYCHY # ls -la /dev/sharevg2
crw-rw---- 1 root      system      35,  0 Jul 20 18:40
/dev/sharevg2
```

4. Create a logical volume and file system.

When you create a journaled file system, AIX creates the corresponding logical volume. Therefore, you do not need to define a logical volume.

Run **lspv <disk name>** to determine how much free space you can use. As shown in Example 2-8, there is 10,224 MB of free space to use.

Example 2-8 Check free space

```
(0) root@TYCHY # lspv hdisk2
PHYSICAL VOLUME:    hdisk2                      VOLUME GROUP:
sharevg2
PV IDENTIFIER:      00cca5e4a543407d VG IDENTIFIER
00cca5e400004c0000000138a54340ba
PV STATE:           active
STALE PARTITIONS:   0                           ALLOCATABLE:     yes
PP SIZE:            16 megabyte(s)             LOGICAL VOLUMES: 0
TOTAL PPs:          639 (10224 megabytes)    VG DESCRIPTORS: 2
FREE PPs:           639 (10224 megabytes)    HOT SPARE:      no
USED PPs:           0 (0 megabytes)           MAX REQUEST:  256
kilobytes
FREE DISTRIBUTION: 128..128..127..128..128
USED DISTRIBUTION: 00..00..00..00..00
MIRROR POOL:       None
```

Create two file systems on this disk. You can create a file system using either **smit** or the command-line interface (CLI). In our scenario, we create two file systems:

- /home/db2inst/db2home
- /home/db2inst/db2

To use **smit** in the graphical mode, you must set up the DISPLAY system variable first (Example 2-9).

Example 2-9 Set display

```
(0) root@TYCHY # export DISPLAY=localhost:10.0
(0) root@TYCHY # echo $DISPLAY
localhost:10.0
```

If you connect to Linux through SSH, you must use the **-X** option (Example 2-10).

Example 2-10 SSH connection with X forwarding

```
ssh -X 9.156.47.80 -l root
```

If you connect to PuTTY through SSH, you must enable X11 forwarding and set the X display location. You can find those options by selecting Connection → SSH → X11.

Under Windows OS, you must also install an X Window System server, such as Xming.

Example 2-11 Sample configuration for X Window System server

AIX server:

```
(0) root@TYCHY # export DISPLAY=localhost:10.0
```

Xming server:

```
multiple windows
display number = 10
no access control - checked
```

Putty:

```
Session > Connection type: SSH
connection > ssh > X11: Enable X11 forwarding- checked
connection > ssh > X11 > X display location: localhost:10.0
```

As the root user, run **smit crfs**, and select Add an Enhanced Journaled → File System Add an Enhanced Journaled File System → sharevg2.

Set the fields Number of units and MOUNT POINT. As shown in Figure 2-7, set MOUNT POINT to home/db2inst/db2home. If you want to place your second file system on the same share disk as we did, do not use the whole empty space for the first file system.

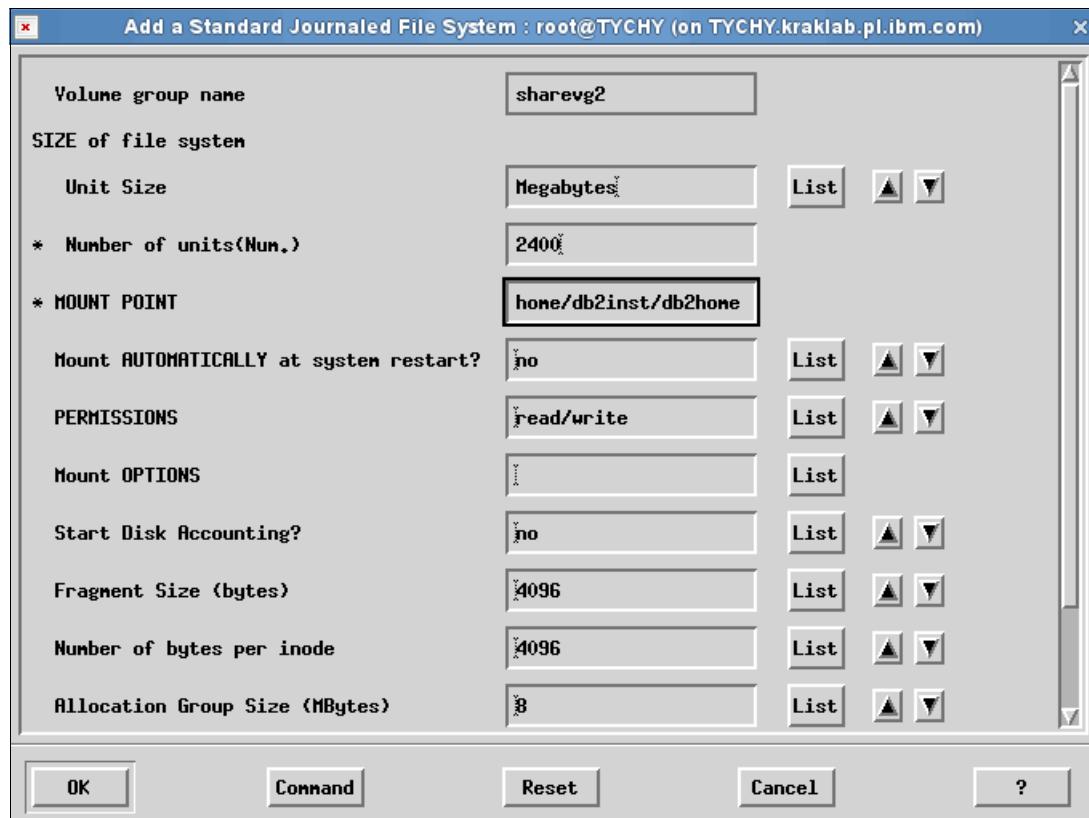


Figure 2-7 Create a file system with smit

When you click **Select**, the status window opens and shows the command execution status.

In our scenario, the second file system is created with the CLI (Example 2-12). In the example, both file systems are also checked and listed.

Example 2-12 Create a file system with the CLI

```
(0) root@TYCHY # crfs -v jfs2 -m /home/db2inst/db2 -g sharevg2 -a  
size=7200M  
File system created successfully.  
7372368 kilobytes total disk space.
```

```

New File System size is 14745600
(0) root@TYCHY # fsck /home/db2inst/db2
The current volume is: /dev/fslv02
Primary superblock is valid.
*** Phase 1 - Initial inode scan
*** Phase 2 - Process remaining directories
*** Phase 3 - Process remaining files
*** Phase 4 - Check and repair inode allocation map
*** Phase 5 - Check and repair block allocation map
File system is clean.
(0) root@TYCHY # fsck /home/db2inst/db2home
The current volume is: /dev/fslv02
Primary superblock is valid.
*** Phase 1 - Initial inode scan
*** Phase 2 - Process remaining directories
*** Phase 3 - Process remaining files
*** Phase 4 - Check and repair inode allocation map
*** Phase 5 - Check and repair block allocation map
File system is clean.
(0) root@TYCHY # lsfs | grep db2inst/
/dev/fslv02      --          /home/db2inst/db2home  jfs2  4915200 --
no   no
/dev/fslv03      --          /home/db2inst/db2      jfs2  14745600 --
no   no

```

- Determine if the same major number shown in Example 2-7 on page 38 is available on the failover node. Run the following command to determine the available major numbers:

(F) # lvolstmajor

Example 2-13 shows the output of the **lvolstmajor** command. This output means that the failover node can use the volume group major number “35” for “sharevg2”.

Example 2-13 Sample output of the lvolstmajor command

(F) root@KRAKOW # lvolstmajor
35...

- Import the volume group from the owner node to the failover node.

After you create the logical volumes on the owner node, you must refresh the volume group’s device files on the failover node. The device files are special files that are stored in the /dev directory and are used for device control. Use the **exportvg** and **importvg** commands to refresh these device files.

To import volume group and file system settings from the owner node to the failover node, run the following commands:

- a. **varyoffvg** and **exportvg** on the owner node
- b. **rmdev**, **cfgmgr**, and **importvg** on the failover node

Example 2-14 shows how to move volume group settings from the owner node to the failover node.

Example 2-14 Export and import logical volume settings

```
(0) root@TYCHY # lspv | grep hdisk2
hdisk2          00cca5e4a543407d           sharevg2
active
(0) root@TYCHY # varyoffvg sharevg2
(0) root@TYCHY # exportvg sharevg2
(0) root@TYCHY # lspv | grep hdisk2
hdisk2          00cca5e4a543407d           None
(F) root@KRAKOW # lspv | grep hdisk2
hdisk2          none                      None
(F) root@KRAKOW # rmdev -l hdisk1 -Rd
hdisk1 deleted
(F) root@KRAKOW # cfgmgr
(F) root@KRAKOW # lspv | grep hdisk2
hdisk2          00cca5e4a543407d           None
```

Import the volume group, specifying the required major numbers as follows:

```
(F) #importvg -y <volume group name> -V <first major number>
<pv name>
```

Example 2-15 shows how to import the volume group sharevg2.

Example 2-15 Sample output of the importvg command

```
(F) root@KRAKOW # lspv |grep hdisk2
hdisk2          00cca5e4a543407d           None
(F) root@KRAKOW # importvg -y sharevg2 -V 35 hdisk2
sharevg2
(F) root@KRAKOW # lspv |grep hdisk2
hdisk2          00cca5e4a543407d           sharevg2      active
(F) root@KRAKOW # lsfs
...
/dev/fs1v02     --          /home/db2inst/db2home  jfs2  4915200 --
no   no
/dev/fs1v03     --          /home/db2inst/db2      jfs2  14745600 --
no   no
```

Example 2-16 shows how to make hdisk2 active again on the owner node.

Example 2-16 Make share disk active

```
(F) root@KRAKOW # varyoffvg sharevg2
(0) root@TYCHY # importvg -y sharevg2 -V 35 hdisk2
```

Row devices: 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 must change them manually.

Creating a DB2 instance

To create a DB2 instance on a shared disk, complete the following steps:

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 must create a DB2 instance owner user and you might need to create a group. The DB2 instance owner should have the same user ID and group ID on all the nodes in the cluster domain. The DB2 instance owner should have the same password on all cluster nodes.

Example 2-17 shows how to create an instance owner user with a home directory on the share disk. The example assumes that the db2iadm1 group exists and shows that the user ID of DB2 instance owner db2inst is 216 and its group ID is 102. Ensure that they are same in every node.

Example 2-17 Create an instance owner

```
(0) root@TYCHY # mount /home/db2inst/db2home
(0) root@TYCHY # useradd -d /home/db2inst/db2home -g db2iadm1
db2inst
(0) root@TYCHY # lsuser -a id pggrp db2inst
db2inst id=216 pggrp=db2iadm1
(0) root@TYCHY # lsgroup -a id db2iadm1
db2iadm1 id=102
(0) root@TYCHY # chown db2inst:db2iadm1 /home/db2inst/db2home
(0) root@TYCHY # lsf | grep /home/db2inst/db2home/dev/fs1v02 -- 
/home/db2inst/db2home jfs2 4915200 -- no no
(0) root@TYCHY # chown db2inst:db2iadm1 /dev/fs1v02
```

Example 2-18 shows how to create an instance owner user on a failover node.

Example 2-18 Create an instance owner user on a failover node

```
(F) root@KRAKOW # useradd -d /home/db2inst/db2home -g db2iadml -u  
216 db2inst  
(F) root@KRAKOW # lsuser -a id pgrp db2inst  
db2inst id=216 pgrp=db2iadml
```

2. Create the DB2 instance.

Run **db2icrt** to create the DB2 instance. The command is in the instance directory on the DB2 installation path.

```
(0) # <DB2 install path>/instance/db2icrt -u <fenced user name>  
<instance owner name>
```

Example 2-19 shows how to create the DB2 instance.

Example 2-19 DB2 instance creation

```
(0) root@TYCHY # cd /opt/IBM/db2/V10.1/instance  
(0) root@TYCHY # ./db2icrt -u db2fenc1 db2inst  
DBI1446I The db2icrt command is running, please wait.  
DB2 installation is being initialized.  
  
Total number of tasks to be performed: 4  
Total estimated time for all tasks to be performed: 309 second(s)  
  
Task #1 start  
Description: Setting default global profile registry variables  
Estimated time 1 second(s)  
Task #1 end  
  
Task #2 start  
Description: Initializing instance list  
Estimated time 5 second(s)  
Task #2 end  
  
Task #3 start  
Description: Configuring DB2 instances  
Estimated time 300 second(s)  
Task #3 end  
  
Task #4 start  
Description: Updating global profile registry  
Estimated time 3 second(s)  
Task #4 end  
  
The execution completed successfully.
```

```
For more information see the DB2 installation log at  
"/tmp/db2icrt.log.11403434".  
DBI1070I Program db2icrt completed successfully.  
(0) root@TYCHY # ./db2ilist  
db2inst
```

Example 2-20 shows how to change the password for the instance owner.

Example 2-20 Change the password

```
(0) root@TYCHY # passwd -a db2inst  
(F) root@KRAKOW # passwd -a db2inst
```

There is a file named .profile in /home/db2inst/db2home. This file was created during user creation. Later on, during instance creation, the .profile file was updated. This file should contain a call to the db2profile procedure (Example 2-21).

Example 2-21 Content of .profile

```
(0) root@TYCHY # cd /home/db2inst/db2home  
(0) root@TYCHY # cat .profile  
...  
# The following three lines have been added by IBM DB2 instance  
utilities.  
if [ -f /home/db2inst/db2home/sql1lib/db2profile ]; then  
    . /home/db2inst/db2home/sql1lib/db2profile  
fi  
...
```

Example 2-22 shows how to configure TCP/IP communication. Create a connection and log on as the db2inst user. You might be asked to change your password the first time you do this task.

Example 2-22 Configuration of TCP/IP communication

```
(0) db2inst@TYCHY $ db2start  
07/22/2012 23:30:12      0  0   SQL1063N  DB2START processing was  
successful.  
SQL1063N  DB2START processing was successful.  
(0) db2inst@TYCHY $ db2set db2comm=tcpip  
(0) db2inst@TYCHY $ db2 update dbm cfg using svcename DB2_db2inst  
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command completed  
successfully.  
(0) db2inst@TYCHY $ cat /etc/services |grep DB2_db2inst  
DB2_db2inst      60012/tcp
```

```
DB2_db2inst_1  60013/tcp
DB2_db2inst_2  60014/tcp
DB2_db2inst_END 60015/tcp
(0) db2inst@TYCHY $ db2stop
07/22/2012 23:46:43      0 0  SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
(0) db2inst@TYCHY $ db2start
07/22/2012 23:47:10      0 0  SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
```

3. Create the SAMPLE database:

Create the SAMPLE database to test the cluster domain in the /home/db2inst/db2 directory (Example 2-23).

Example 2-23 Create SAMPLE database

```
(0) root@TYCHY # mount /home/db2inst/db2
(0) root@TYCHY # chown db2inst:db2iadml /home/db2inst/db2
(0) db2inst@TYCHY $ db2 create database sample on /home/db2inst/db2
DB20000I The CREATE DATABASE command completed successfully.
(0) db2inst@TYCHY $ mkdir /home/db2inst/db2/actlog
(0) db2inst@TYCHY $ mkdir /home/db2inst/db2/arclog
(0) db2inst@TYCHY $ db2 update db cfg for sample using NEWLOGPATH
/home/db2inst/db2/actlog
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
(0) db2inst@TYCHY $ db2 update db cfg for sample using LOGARCHMETH1
disk:/home/db2inst/db2/arclog
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
(0) db2inst@TYCHY $ db2 backup db sample to /dev/null
```

Backup successful. The timestamp for this backup image is : 20120722235729

```
(0) db2inst@TYCHY $ db2 connect to sample
```

Database Connection Information

```
Database server      = DB2/AIX64 10.1.0
SQL authorization ID = DB2INST
Local database alias = SAMPLE
```

Configuring shared disk HA: If you want to configure shared disk HA on an existing DB2 instance, you must move DB2 objects to the shared volume group. DB2 objects include the instance home directory, diagpath, and audit path, and all the database objects.

4. 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 to run the DB2 instance correctly. The **db2icrt** command adds the entries to the /etc/services file on the owner node automatically, but you must add them on the failover node manually. Example 2-24 shows the sample of node1 and node2.

Example 2-24 The entries of /etc/services file on owner and failover nodes

```
(0) root@TYCHY $ cat /etc/services |grep db2inst
```

```
DB2_db2inst    60012/tcp
DB2_db2inst_1  60013/tcp
DB2_db2inst_2  60014/tcp
DB2_db2inst_END 60015/tcp
```

```
(F) root@KRAKOW # cat /etc/services |grep db2inst
```

```
DB2_db2inst    60012/tcp
DB2_db2inst_1  60013/tcp
DB2_db2inst_2  60014/tcp
DB2_db2inst_END 60015/tcp
```

5. Confirm that the DB2 instance can move to another node manually.

If the DB2 instances are not configured properly, the cluster domain might have unexpected behavior after you include the DB2 instances.

Example 2-25 shows the process to shut down the owner node.

Example 2-25 Shutdown process on owner node “TYCHY”

```
(0) db2inst@TYCHY $ db2_ps
```

```
Node 0
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
db2inst	9699342	14352570	1	23:47:09	-	0:33	db2sysc 0
root	6815794	9699342	0	23:47:09	-	0:00	db2ckpwd 0
root	7864512	9699342	0	23:47:09	-	0:00	db2ckpwd 0
root	10092640	9699342	0	23:47:09	-	0:00	db2ckpwd 0

```
(0) db2inst@TYCHY $ db2stop
```

```
07/23/2012 09:22:20      0  0  SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
```

```
(0) db2inst@TYCHY $ db2_ps
```

```
Node 0
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
-----	-----	------	---	-------	-----	------	-----

```
(0) db2inst@TYCHY $ exit
```

```
(0) root@TYCHY # df |grep db2inst
```

/dev/fs1v02	4915200	3912760	21%	209	1%	/home/db2inst/db2home
/dev/fs1v03	14745600	14220368	4%	84	1%	/home/db2inst/db2

```
(0) root@TYCHY # umount /home/db2inst/db2home
(0) root@TYCHY # umount /home/db2inst/db2
(0) root@TYCHY # df |grep db2inst
(0) root@TYCHY # lspv |grep sharevg2
hdisk2          00cca5e4a543407d           sharevg2      active
(0) root@TYCHY # varyoffvg sharevg2
(0) root@TYCHY # lspv |grep sharevg2
hdisk2          00cca5e4a543407d           sharevg2
```

6. The db2nodes.cfg file and the starting instance on the failover node.

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 must change the host name in the db2nodes.cfg file or have DB2 change the file by starting the DB2 instance with **db2gcf**. In our example, we use **db2gcf** during the instance start operation.

Example 2-26 shows the process to start DB2 on the failover node.

Remember to ensure that the /etc/services files on both nodes have the same entries for the DB2 instance.

Example 2-26 Start process on failover node “KRAKOW”

```
(F) root@KRAKOW # lspv |grep sharevg2
hdisk2          00cca5e4a543407d           sharevg2
(F) root@KRAKOW # varyonvg sharevg2
(F) root@KRAKOW # lspv |grep sharevg2
hdisk2          00cca5e4a543407d           sharevg2
active
(F) root@KRAKOW # mount /home/db2inst/db2home
(F) root@KRAKOW # mount /home/db2inst/db2
(F) root@KRAKOW # exit
(F) db2inst@KRAKOW # echo "0 $(hostname) 0 $(hostname)
0 KRAKOW.kraklab.pl.ibm.com 0 KRAKOW.kraklab.pl.ibm.com
(F) db2inst@KRAKOW # echo "0 $(hostname) 0 $(hostname)
>/sqllib/db2nodes.cfg
(F) db2inst@KRAKOW # db2gcf -u -i db2inst

Instance : db2inst
DB2 Start : Success
```

In our scenario, we manually move an instance to failover node and now we have a situation where:

- The owner node becomes the failover node.
- The failover node becomes the owner node.

Example 2-27 shows how to switch the owner and failover nodes back.

Example 2-27 Switch owner and failover nodes

```
(F) db2inst@KRAKOW $ db2stop
(F) db2inst@KRAKOW $ db2_ps
(F) db2inst@KRAKOW $ exit
(F) root@KRAKOW # umount /home/db2inst/db2home
(F) root@KRAKOW # umount /home/db2inst/db2
(F) root@KRAKOW # varyoffvg sharevg2
(0) root@TYCHY # varyonvg sharevg2
(0) root@TYCHY # mount /home/db2inst/db2home
(0) root@TYCHY # mount /home/db2inst/db2
(0) db2inst@TYCHY $ db2gcf -s -i db2inst
(0) db2inst@TYCHY $ echo "0 $(hostname)" 0 "$(hostname)"
>/sqllib/db2nodes.cfg
(0) db2inst@TYCHY $ db2gcf -u -i db2inst
```

Cluster preparation

Two tasks on the nodes are required before you start node clustering:

1. Preparing the nodes to use **db2haicu**.

Before you use the **db2haicu** utility, run **preprnode** on every node that is a part of the cluster domain. The command syntax is:

```
(0)(F) # preprnode <node name> <node name>...
```

2. In our environment, as a root user, we issue the command that is shown in Example 2-28 on both nodes, TYCHY and KRAKOW. 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 2-28 Sample command of preprnode command

```
(0) root@TYCHY # preprnode TYCHY KRAKOW
(F) root@KRAKOW # preprnode TYCHY KRAKOW
```

System clock synchronization

Synchronizing the date and time among nodes on a cluster is not required, but do this task, because synchronized clocks can make your problem determination more straightforward. When you analyze the logs on different nodes, you can see the time sequence of the events without any adjustment. You can use the network time protocol (NTP) for this purpose. For more information about how to configure NTP for your system, see your operating system documentation.

Creating an XML configuration file for db2haicu

Example 2-29 shows a sample XML file for the DB2 shared storage HA configuration that is used in our scenario. This XML file contains all the information that **db2haicu** needs.

Example 2-29 XML configuration file for db2haicu

```
<?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="db2HAdomain">
        <Quorum quorumDeviceProtocol="network"
quorumDeviceName="9.156.47.82"/>
        <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
            <Interface interfaceName="en0" clusterNodeName="TYCHY">
                <IPAddress baseAddress="9.156.47.80" subnetMask="255.255.255.0"
networkName="db2_public_network_0"/>
            </Interface>
            <Interface interfaceName="en0" clusterNodeName="KRAKOW">
                <IPAddress baseAddress="9.156.47.81" subnetMask="255.255.255.0"
networkName="db2_public_network_0"/>
            </Interface>
        </PhysicalNetwork>
        <ClusterNode clusterNodeName="TYCHY"/>
        <ClusterNode clusterNodeName="KRAKOW"/>
    </ClusterDomain>
    <FailoverPolicy>
        <Mutual></Mutual>
    </FailoverPolicy>
    <DB2PartitionSet>
        <DB2Partition dbpartitionnum="0" instanceName="db2inst">
            <VirtualIPAddress baseAddress="9.156.47.199"
subnetMask="255.255.255.0" networkName="db2_public_network_0"/>
            <MutualPair systemPairNode1="TYCHY" systemPairNode2="KRAKOW" />
        </DB2Partition>
    </DB2PartitionSet>
    <HADBSet instanceName="db2inst">
        <HADB databaseName="SAMPLE" />
    </HADBSet>
</DB2Cluster>
```

Where:

- ▶ The <ClusterDomain> element

This element covers all cluster-wide information, including quorum, cluster node, and cluster domain name.

The <PhysicalNetwork> subelement has all network-related information, including the network name and the network interface.

You must define all network interfaces in this element whether you use the definition or not. 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 an 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 that is associated with the instance.

The <MutualPair> is the node pair that is joined to the cluster domain. The systemPairNode1 attribute is the active (owner) node, and the systemPairNode2 attribute 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.

Setting up a cluster domain using db2haicu with an XML file

Before you run **db2haicu**, ensure that the DB2 instance is started. **db2haicu** needs a running DB2 instance.

Running db2haicu: Run **db2haicu** on the owner node. The *owner node* here is the node that is specified in the systemPairNode1 attribute. If you run **db2haicu** on the failover node (the node that is specified in the systemPairNode2 attribute), Tivoli SA MP tries to take over the DB2 resource group to the owner node as soon as **db2haicu** execution is complete.

Run **db2haicu** as the DB2 instance owner. Here is the **db2haicu** command syntax:

(0) \$ db2haicu -f <XML file name>

Example 2-30 shows a successful completion of the command execution.

Example 2-30 Sample output of db2haicu execution using XML setup up mode

```
(0) db2inst@TYCHY $ db2haicu -f db2ha_sharedstorage_mutual_.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 'db2inst'. The cluster configuration that follows apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu need to activate all databases for the instance to discover all paths ...

Creating domain 'db2HAdomain' in the cluster ...

Creating domain 'db2HAdomain' in the cluster was successful.

Configuring quorum device for domain 'db2HAdomain' ...

Configuring quorum device for domain 'db2HAdomain' was successful.

Adding network interface card 'en0' on cluster node 'TYCHY' to the network 'db2_public_network_0' ...

Adding network interface card 'en0' on cluster node 'TYCHY' to the network 'db2_public_network_0' was successful.

Adding network interface card 'en0' on cluster node 'KRAKOW' to the network 'db2_public_network_0' ...

Adding network interface card 'en0' on cluster node 'KRAKOW' to the network 'db2_public_network_0' 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 Tivoli SA MP **issam** command and the DB2 command **db2pd** are useful for verifying DB2 clustering with Tivoli SA MP. **1ssam** lists all the resource groups and resources that are contained in the resource groups and the information about operational states of these objects. The command syntax is as follows:

```
(0) (F) # 1ssam
```

Example 2-31 shows that the resource group, db2_db2inst_0-rg, is online and the active node is TYCHY, which owns all resources in this cluster domain.

Example 2-31 Sample output of 1ssam command

```
(0) db2inst@TYCHY $ 1ssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst_0-rs
        |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
        '- Online IBM.Application:db2_db2inst_0-rs:TYCHY
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
        |- Offline
IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
    '- Online
IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
        |- Offline
IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
    '- Online
IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
    '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs
        |- Offline IBM.ServiceIP:db2ip_9_156_47_199-rs:KRAKOW
        '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
    |- Online IBM.PeerNode:TYCHY:TYCHY
    '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:en0:TYCHY
    '- Online IBM.NetworkInterface:en0:KRAKOW
```

In DB2 10, the **db2pd** command option **-ha** provides data of the cluster domain:

```
db2pd -ha
```

Example 2-32 shows the output of this command.

Example 2-32 Sample output for the db2pd -ha command

```
(0) db2inst@TYCHY $ db2pd -ha
      DB2 HA Status
      Instance Information:
      Instance Name          = db2inst
      Number Of Domains     = 1
      Number Of RGs for instance = 1

      Domain Information:
      Domain Name            = db2HAdomain
      Cluster Version         = 3.1.2.2
      Cluster State           = Online
      Number of nodes         = 2

      Node Information:
      Node Name               State
      -----                  -----
      TYCHY                  Online
      KRAKOW                 Online

      Resource Group Information:
      Resource Group Name     = db2_db2inst_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
      -----
      TYCHY
      KRAKOW

      Member Resource Information:
      Resource Name           = db2mnt-home_db2inst_db2-rs
      Resource State          = Online
      Resource Type            = Mount
      Mount Resource Path      = /home/db2inst/db2
      Number of Allowed Nodes = 2
      Allowed Nodes
      -----
      TYCHY
      KRAKOW
```

Resource Name	= db2ip_9_156_47_199-rs
Resource State	= Online
Resource Type	= IP
Resource Name	= db2_db2inst_0-rs
Resource State	= Online
Resource Type	= DB2 Member
DB2 Member Number	= 0
Number of Allowed Nodes	= 2
Allowed Nodes	

TYCHY	
KRAKOW	
Resource Name	= db2mnt-home_db2inst_db2home-rs
Resource State	= Online
Resource Type	= Mount
Mount Resource Path	= /home/db2inst/db2home
Number of Allowed Nodes	= 2
Allowed Nodes	

TYCHY	
KRAKOW	
Network Information:	
Network Name	Number of Adapters

db2_public_network_0	2
Node Name	Adapter Name

TYCHY.kraklab.pl.ibm.com	en0
KRAKOW.kraklab.pl.ibm.com	en0
Quorum Information:	
Quorum Name	Quorum State

Fail	Offline
db2_Quorum_Network_9_156_47_82:16_6_29	Online
Operator	Offline

2.5 Administration

You must perform regularly some maintenance tasks on the node in the shard disk HA cluster domain, such as upgrading software. This section demonstrates how to detach one node from the cluster domain and how to stop the cluster domain for system maintenance.

2.5.1 The node maintenance scenario

You can use the idle state of the failover node to perform the maintenance tasks on that node without stopping the entire cluster domain. In this process, you detach the failover node from the cluster, perform the maintenance work, and then reattach the node back to the cluster. If the maintenance work is performed on the owner node, you must swap the role of the node first.

Complete the following steps:

1. Swap the owner node.

If the node to be maintained is the owner node, switch its role to 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 by running **lssam**.

2. Detach the failover node.

Run **samctrl** to detach the maintenance node:

```
(0) # samctrl -u a <node to detach>
```

Example 2-33 shows how to swap the owner node and detach it for maintenance.

Example 2-33 Sample of the samctrl command and its output

```
(0) root@TYCHY # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
    '- Online IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
    '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
    '- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
```

```

        '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs
            |- Offline IBM.ServiceIP:db2ip_9_156_47_199-rs:KRAKOW
            '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs:TYCHY
    Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
        |- Online IBM.PeerNode:TYCHY:TYCHY
        '- Online IBM.PeerNode:KRAKOW:KRAKOW
    Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:en0:TYCHY
        '- Online IBM.NetworkInterface:en0:KRAKOW
(0) root@TYCHY # rgreq -o move db2_db2inst_0-rg
Action on resource group "db2_db2inst_0-rg" returned Token
"0x4a6b13a868cf47e3500ea940003b95b" .
(0) root@TYCHY # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst_0-rs
        |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
        '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
        |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
        '- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
        |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
        '- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
    '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs
        |- Online IBM.ServiceIP:db2ip_9_156_47_199-rs:KRAKOW
        '- Offline IBM.ServiceIP:db2ip_9_156_47_199-rs:TYCHY
    Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
        |- Online IBM.PeerNode:TYCHY:TYCHY
        '- Online IBM.PeerNode:KRAKOW:KRAKOW
    Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:en0:TYCHY
        '- Online IBM.NetworkInterface:en0:KRAKOW
(0) root@TYCHY # samctrl -u a TYCHY
(0) root@TYCHY # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst_0-rs
        |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
        '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY Node=Excluded
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
        |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
        '- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
Node=Excluded
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
        |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW

```

```

      '- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
Node=Excluded
      '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs
          |- Online IBM.ServiceIP:db2ip_9_156_47_199-rs:KRAKOW
          '- Offline IBM.ServiceIP:db2ip_9_156_47_199-rs:TYCHY Node=Excluded
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
      |- Online IBM.PeerNode:TYCHY:TYCHY Node=Excluded
      '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
      |- Online IBM.NetworkInterface:en0:TYCHY Node=Excluded
      '- Online IBM.NetworkInterface:en0:KRAKOW

```

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 the OS or upgrade software.

Example 2-34 shows the resource status when the owner node is down.

Example 2-34 Resource status when the owner node is turned off

```

(F) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Control=MemberInProblemState Nominal=Online
    |- Online IBM.Application:db2_db2inst_0-rs Control=MemberInProblemState
        |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
        '- Failed offline IBM.Application:db2_db2inst_0-rs:TYCHY Node=Offline
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs Control=MemberInProblemState
        |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
        '- Failed offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY Node=Offline
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs Control=MemberInProblemState
        |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
        '- Failed offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
Node=Offline
    '- Online IBM.ServiceIP:db2ip_9_156_47_199-rs Control=MemberInProblemState
        |- Online IBM.ServiceIP:db2ip_9_156_47_199-rs:KRAKOW
        '- Failed offline IBM.ServiceIP:db2ip_9_156_47_199-rs:TYCHY Node=Offline
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
    |- Offline IBM.PeerNode:TYCHY:TYCHY Node=Offline
    '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
    |- Offline IBM.NetworkInterface:en0:TYCHY Node=Offline
    '- Online IBM.NetworkInterface:en0:KRAKOW

```

Node status: When you shut down the RSCT or Tivoli SA MP on the detached node, the output of the `lssam` command shows Failed Offline for the node. There is no impact to the cluster node, but the output is changed.

4. Reintegration in to the cluster domain.

When you want to reintegrate the detached node, run **samctrl** again:

```
(0) # samctrl -u d <node to detach>
```

After you run **samctrl**, ensure that there is no Excluded mode in the output of the **lssam** command. Example 2-35 shows how to run **samctrl**.

Example 2-35 Attaching an owner node back to the cluster and swapping the owner node to be back online

```
(F) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
    '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY Node=Excluded
...
(F) root@KRAKOW # samctrl -u d TYCHY
(F) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
    '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
...
(F) root@KRAKOW # rgreq -o move db2_db2inst_0-rg
```

2.6 Cluster maintenance

When you must work on DB2 instances or all the resources, you must stop the cluster domain. For example, you must stop the cluster domain when you migrate DB2 to a later version. Run **chrg** and **stoprpdomain** to stop the cluster domain.

To stop the cluster domain, complete the following steps:

1. Change the nominal state of the resource groups to offline.

Changing the nominal status of the resource group to *offline* causes Tivoli SA MP to try to stop the all resources in the resource group. After that, you can stop the cluster domain. Run the following command to make this change:

```
(0) # chrg -o offline <resource group>
```

After you run **chrg**, ensure that all resource group and resources in this domain are offline by running **lssam**.

2. Stop the cluster domain.

Run **stoprpdomain** to stop the cluster domain:

```
(0) # stoprpdomain <domain>
```

Example 2-36 shows that the shared_disk_domain cluster domain is stopped.

Example 2-36 Sample command of stopping the cluster domain

```
(0) root@TYCHY # chrg -o offline db2_db2inst_0-rg
(0) root@TYCHY # lssam
Offline IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Offline
  |- Offline IBM.Application:db2_db2inst_0-rs
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
      '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
...
(0) root@TYCHY # lsrpdomain
Name          OpState RSCTActiveVersion MixedVersions TSPort GSPort
db2HAdomain  Online   3.1.2.2           No        12347  12348
(0) root@TYCHY # stoprpdomain db2HAdomain
(0) root@TYCHY # lsrpdomain
Name          OpState RSCTActiveVersion MixedVersions TSPort GSPort
db2HAdomain  Offline  3.1.2.2           No        12347  12348
```

3. Perform maintenance.

You can perform maintenance tasks on these nodes, for example, to migrate the DB2 or Tivoli SA MP.

4. Start the cluster domain.

After the maintenance, restart the cluster domain by running **startrpdomain**:

```
(0) # startrpdomain <domain>
```

5. Change the nominal state of the resource groups to online.

Run **chrg** to change the nominal state of the resource groups:

```
(0) # chrg -o online <resource group>
```

After you run **chrg**, verify that the resource groups and all resources in this domain are online.

Example 2-37 shows that the db2HAdomain is back online.

Example 2-37 Sample command of stopping the cluster domain

```
(0) root@TYCHY # startrpdomain db2HAdomain
(0) root@TYCHY # lsrpdomain
Name          OpState RSCTActiveVersion MixedVersions TSPort GSPort
```

```
db2HAdomain Online 3.1.2.2           No          12347 12348
(0) root@TYCHY # chrg -o online db2_db2inst_0-rg
(0) root@TYCHY # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
| - Online IBM.Application:db2_db2inst_0-rs
|   | - Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
|   | - Online IBM.Application:db2_db2inst_0-rs:TYCHY
...

```

2.6.1 Deleting a domain

If you want to delete resource groups for the current database manager instance, run the following command:

```
(0) (F) # db2haicu -delete <resource group>
```

If there are no resource groups that are left in the cluster domain after **db2haicu** removes the resource groups, then **db2haicu** also removes the cluster domain.

Running **db2haicu** with the **-delete** parameter causes the current database manager instance to cease to be configured for high availability. If the database manager instance is no longer configured for high availability, then the database manager does not coordinate with the cluster manager if you perform any database manager administrative operations that require related cluster configuration changes.

To reconfigure a database manager instance for high availability, you can run **db2haicu** again.

Example 2-38 shows how to delete a high availability configuration and create it one more time from an XML file.

Example 2-38 Command to delete a high availability configuration

```
(0) db2inst@TYCHY $ 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 'db2inst'. The cluster configuration that follows 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 machine for an existing active cluster domain ...

db2haicu found a cluster domain called 'db2HAdomain' on this machine. The cluster configuration that follows apply to this domain.

Removing DB2 database partition '0' from the cluster ...

Removing DB2 database partition '0' from the cluster was successful.

Deleting the domain 'db2HAdomain' from the cluster ...

Deleting the domain 'db2HAdomain' from the cluster was successful.

All cluster configurations have been completed successfully. db2haicu exiting ...

(0) db2inst@TYCHY \$ db2haicu -f db2ha_sharedstorage_mutual_.xml

2.7 Testing

This section shows some resource failure tests that you can use to verify the HA cluster setup:

- ▶ Operating system failure
- ▶ Power failure
- ▶ Network failure
- ▶ DB2 instance failure

We suggest that you perform similar tests before you deploy the HA configuration to production.

2.7.1 Operating system failure

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. 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 owner node failure.

Example 2-39 shows the state of the cluster domain before shutdown starts. TYCHY is online and owns the resources.

Example 2-39 Cluster domain state before shutdown starts

```
(0) db2inst@TYCHY $ lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
    '- Online IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
    '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
    '- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs
    |- Offline IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW
    '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
  '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:en0:TYCHY
  '- Online IBM.NetworkInterface:en0:KRAKOW
```

The **shutdown -Fr** command is run on the TYCHY owner node to simulate an operating system failure. Example 2-40 shows that the failover is taking place at the cluster domain and starting a failover after the shutdown starts.

Example 2-40 Failover is taking place

```
(0) root@TYCHY # shutdown -Fr
(0) root@KRAKOW # lssam
'Pending online IBM.ResourceGroup:db2_db2inst_0-rg Control=MemberInProblemState
Nominal=Online
  |- Pending online IBM.Application:db2_db2inst_0-rs
Control=MemberInProblemState
  |- Pending online IBM.Application:db2_db2inst_0-rs:KRAKOW
  '- Failed offline IBM.Application:db2_db2inst_0-rs:TYCHY Node=Offline
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
Control=MemberInProblemState
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
  '- Failed offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
Node=Offline
```

```
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs  
Control=MemberInProblemState  
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW  
' - Failed offline  
IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY Node=Offline  
' - Online IBM.ServiceIP:db2ip_9_156_47_191-rs Control=MemberInProblemState  
| - Online IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW  
' - Failed offline IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY  
Node=Offline  
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ  
| - Offline IBM.PeerNode:TYCHY:TYCHY Node=Offline  
' - Online IBM.PeerNode:KRAKOW:KRAKOW  
Online IBM.Equivalency:db2_public_network_0  
| - Offline IBM.NetworkInterface:en0:TYCHY Node=Offline  
' - Online IBM.NetworkInterface:en0:KRAKOW
```

Example 2-41 shows that Tivoli SA MP failed over the resources to the KRAKOW failover node.

Example 2-41 Failover completed

```
(0) root@KRAKOW # lssam  
Online IBM.ResourceGroup:db2_db2inst_0-rg Control=MemberInProblemState Nominal=Online  
| - Online IBM.Application:db2_db2inst_0-rs Control=MemberInProblemState  
| - Online IBM.Application:db2_db2inst_0-rs:KRAKOW  
' - Failed offline IBM.Application:db2_db2inst_0-rs:TYCHY Node=Offline  
| - Online IBM.Application:db2mnt-home_db2inst_db2-rs Control=MemberInProblemState  
| - Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW  
' - Failed offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY Node=Offline  
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs Control=MemberInProblemState  
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW  
' - Failed offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY  
Node=Offline  
' - Online IBM.ServiceIP:db2ip_9_156_47_191-rs Control=MemberInProblemState  
| - Online IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW  
' - Failed offline IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY Node=Offline  
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ  
| - Offline IBM.PeerNode:TYCHY:TYCHY Node=Offline  
' - Online IBM.PeerNode:KRAKOW:KRAKOW  
Online IBM.Equivalency:db2_public_network_0  
| - Offline IBM.NetworkInterface:en0:TYCHY Node=Offline  
' - Online IBM.NetworkInterface:en0:KRAKOW
```

Example 2-42 shows that the cluster domain completes reintegrating the old owner node.

Example 2-42 Old owner node is reintegrated

```
(0) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
    '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
    '- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
    '- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs
    |- Online IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW
    '- Offline IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
  '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:en0:TYCHY
  '- Online IBM.NetworkInterface:en0:KRAKOW
```

2.7.2 Power failure

The power failure can be simulated by unplugging the power cable. The failover process is similar to the operating system failure. The **lssam** output is similar as well. This test can reassure you that the failover behavior of the cluster domain is as expected.

2.7.3 Network failure

A network failure test checks the HA configuration of the service IP address. 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 en0 network interface in TYCHY.

The next three examples show the transition of the cluster domain upon network failure.

Example 2-43 shows the cluster domain state before the network failure test starts.

Example 2-43 Cluster domain before the network failure test starts

```
(0) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
      '- Online IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
      '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
      '- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs
    |- Offline IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW
      '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
  '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:en0:TYCHY
  '- Online IBM.NetworkInterface:en0:KRAKOW
```

Unplug the network cable. Example 2-44 shows the cluster domain that is starting to fail over the resource from the owner node to the failover node.

Example 2-44 Cluster domain starts the failover process

```
(0) root@KRAKOW # lssam
Pending offline IBM.ResourceGroup:db2_db2inst_0-rg Binding=Sacrificial
Control=MemberInProblemState Nominal=Online
  |- Offline IBM.Application:db2_db2inst_0-rs Binding=Sacrificial
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
      '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Pending offline IBM.Application:db2mnt-home_db2inst_db2-rs
    Binding=Sacrificial
      |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
        '- Pending offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Pending offline IBM.Application:db2mnt-home_db2inst_db2home-rs
    Binding=Sacrificial
      |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
```

```
'- Pending offline
IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Offline IBM.ServiceIP:db2ip_9_156_47_191-rs Binding=Sacrificial
    Control=MemberInProblemState
      |- Offline IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW
        '- Failed offline IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
    '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  |- Offline IBM.NetworkInterface:en0:TYCHY
    '- Online IBM.NetworkInterface:en0:KRAKOW
```

Example 2-45 shows that the cluster domain completes the failover process.
KRAKOW is now the owner node.

Example 2-45 Cluster domain completes the failover process

```
(0) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Control=MemberInProblemState Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Online IBM.Application:db2_db2inst_0-rs:KRAKOW
      '- Offline IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Online IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
      '- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
      '- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs Control=MemberInProblemState
    |- Online IBM.ServiceIP:db2ip_9_156_47_191-rs:KRAKOW
      '- Online IBM.ServiceIP:db2ip_9_156_47_191-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
    '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  |- Offline IBM.NetworkInterface:en0:TYCHY
    '- Online IBM.NetworkInterface:en0:KRAKOW
```

2.7.4 DB2 instance failure

This test checks the DB2 instance failure by stopping the DB2 process. When Tivoli SA MP detects the failure of the resources, Tivoli SA MP tries to start the resource on the same node once. So, 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. Tivoli SA MP detects the failure by the return code (RC) of the monitor script that is run by Tivoli SA MP. Tivoli SA MP acts 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), Tivoli SA MP determines that the node recovered by itself successfully. If the start script returns a non-zero return code (RC<>0) or the next monitor script return code is RC=2 (offline), Tivoli SA MP starts the standby node takeover process immediately. In this test case, the monitor script returns RC=2 (offline), which triggers the Tivoli SA MP to start the failed resource (DB2 processes).

To simulate a DB2 instance failure, stop the DB2 process by running **kill**:

```
(0) # kill <process ID of DB2>
```

The next three examples show the recovery progress of the TYCHY.

Example 2-46 shows the cluster domain state before the DB2 process failed.

Example 2-46 Initial cluster domain state

```
(0) root@KRAKOW # lssam
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst_0-rs
    |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW
      '- Online IBM.Application:db2_db2inst_0-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW
      '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
  |- Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
      '- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_196-rs
    |- Offline IBM.ServiceIP:db2ip_9_156_47_196-rs:KRAKOW
      '- Online IBM.ServiceIP:db2ip_9_156_47_196-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  |- Online IBM.PeerNode:TYCHY:TYCHY
  '- Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
```

```
| - Online IBM.NetworkInterface:en0:TYCHY  
|- Online IBM.NetworkInterface:en0:KRAKOW
```

Stop a DB2 process by running the following commands:

```
► # ps -ef|grep db2sysc |grep db2inst  
► db2inst 10420290 10354752 1 18:13:20 - 0:00 db2sysc 0  
► # kill -9 10420290
```

Example 2-47 shows that the recovery is started on the same node (TYCHY).

Example 2-47 The recovery is started on TYCHY

```
(0) root@KRAKOW # lssam  
Pending online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online  
| - Pending online IBM.Application:db2_db2inst_0-rs  
|   |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW  
|   '- Pending online IBM.Application:db2_db2inst_0-rs:TYCHY  
| - Online IBM.Application:db2mnt-home_db2inst_db2-rs  
|   |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW  
|   '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY  
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs  
|   |- Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW  
|   '- Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY  
'- Online IBM.ServiceIP:db2ip_9_156_47_196-rs  
  |- Offline IBM.ServiceIP:db2ip_9_156_47_196-rs:KRAKOW  
  '- Online IBM.ServiceIP:db2ip_9_156_47_196-rs:TYCHY  
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ  
| - Online IBM.PeerNode:TYCHY:TYCHY  
| '- Online IBM.PeerNode:KRAKOW:KRAKOW  
Online IBM.Equivalency:db2_public_network_0  
| - Online IBM.NetworkInterface:en0:TYCHY  
| '- Online IBM.NetworkInterface:en0:KRAKOW
```

Example 2-48 shows that the restart is successful on TYCHY.

Example 2-48 Self node restart successful

```
(0) root@KRAKOW # lssam  
Online IBM.ResourceGroup:db2_db2inst_0-rg Nominal=Online  
| - Online IBM.Application:db2_db2inst_0-rs  
|   |- Offline IBM.Application:db2_db2inst_0-rs:KRAKOW  
|   '- Online IBM.Application:db2_db2inst_0-rs:TYCHY  
| - Online IBM.Application:db2mnt-home_db2inst_db2-rs  
|   |- Offline IBM.Application:db2mnt-home_db2inst_db2-rs:KRAKOW  
|   '- Online IBM.Application:db2mnt-home_db2inst_db2-rs:TYCHY
```

```
| - Online IBM.Application:db2mnt-home_db2inst_db2home-rs
    | - Offline IBM.Application:db2mnt-home_db2inst_db2home-rs:KRAKOW
    ' - Online IBM.Application:db2mnt-home_db2inst_db2home-rs:TYCHY
  '- Online IBM.ServiceIP:db2ip_9_156_47_196-rs
    | - Offline IBM.ServiceIP:db2ip_9_156_47_196-rs:KRAKOW
    ' - Online IBM.ServiceIP:db2ip_9_156_47_196-rs:TYCHY
Online IBM.Equivalency:db2_db2inst_0-rg_group-equ
  | - Online IBM.PeerNode:TYCHY:TYCHY
  ' - Online IBM.PeerNode:KRAKOW:KRAKOW
Online IBM.Equivalency:db2_public_network_0
  | - Online IBM.NetworkInterface:en0:TYCHY
  ' - Online IBM.NetworkInterface:en0:KRAKOW
```



DB2 and PowerHA SystemMirror

This chapter explains how to integrate DB2 in a PowerHA SystemMirror (formerly known as HACMP) environment. The DB2 in the cluster is in a shared disk. It provides the basic management concepts with considerations to reduce the time that is taken for failover.

This chapter covers the following topics:

- ▶ Overview
- ▶ How DB2 works with PowerHA
- ▶ Planning the PowerHA cluster
- ▶ Setting up the PowerHA cluster
- ▶ Considerations for db2nodes.cfg file
- ▶ Tuning tips for quick failover

3.1 Overview

PowerHA for AIX provides a highly available computing environment. PowerHA 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 PowerHA clusters is to provide a highly available environment for mission-critical applications. In an PowerHA cluster, to ensure the availability of these applications, the applications are placed under PowerHA control.

PowerHA ensures that the applications remain available to client processes even if a component in a cluster fails. To ensure availability if there is a component failure, the PowerHA software moves the application along with resources to another node in the cluster.

Here are some common PowerHA terms:

- ▶ **Topology**

The layout of physical components and connections that are defined in PowerHA.

- ▶ **Cluster**

The group of nodes that work together closely for 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 PowerHA. It might include the IP address that 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 resources 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 a group of storage devices and volume groups that is connected to multiple nodes. Raw devices and file systems that are required to provide services are placed on them. From the perspective of DB2 in a non-HADR implementation of PowerHA, this group would include the database directory, table space containers, and database logs.
- ▶ Application server

The application server is a set of scripts that are used to start and stop the applications that are running on a PowerHA node. One of the scripts is an *application server start script* that starts the applications that are a prerequisite to a provision of client services. The other script is an *application server stop script* that stops applications before you release resource groups. From the perspective of DB2, this set includes scripts to start and stop the database instance.

Although there is only one main script to provide stop and start functionality for all applications and services, these functions can easily be made modular by splitting the tasks into many subscripts, which are called from the main script. For example, with the 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 HACM terminology, see *PowerHA SystemMirror: Master Glossary*, SC23-6757.

3.2 How DB2 works with PowerHA

This section focuses on the PowerHA 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 the database instance on a standby node when the service node has an outage. This classic type of cluster is a *shared disk cluster*. This section explains how DB2 works in a shared disk cluster that is controlled by PowerHA with an example of a simple single partitioned instance.

Figure 3-1 illustrates a PowerHA topology and related DB2 database components. PowerHA manages the group of resources that are required to provide a service, including shared disks where database components are, the IP address for client service, and the application server that handles the starting and stopping of the applications (including the DB2 database instance).

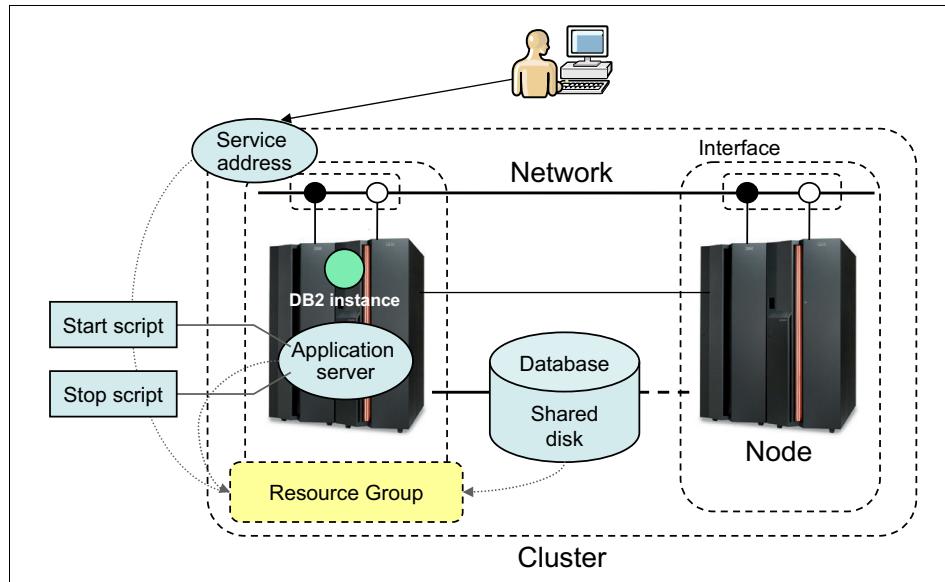


Figure 3-1 PowerHA topology and related DB2 database components

PowerHA continuously sends heartbeats between the nodes in the cluster to identify if and when the other one is down. PowerHA 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 PowerHA 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 action 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 is defined in the application server is issued, which starts the DB2 instance along with other critical services and applications.

Figure 3-2 shows how the resource group was moved to a standby node during failover.

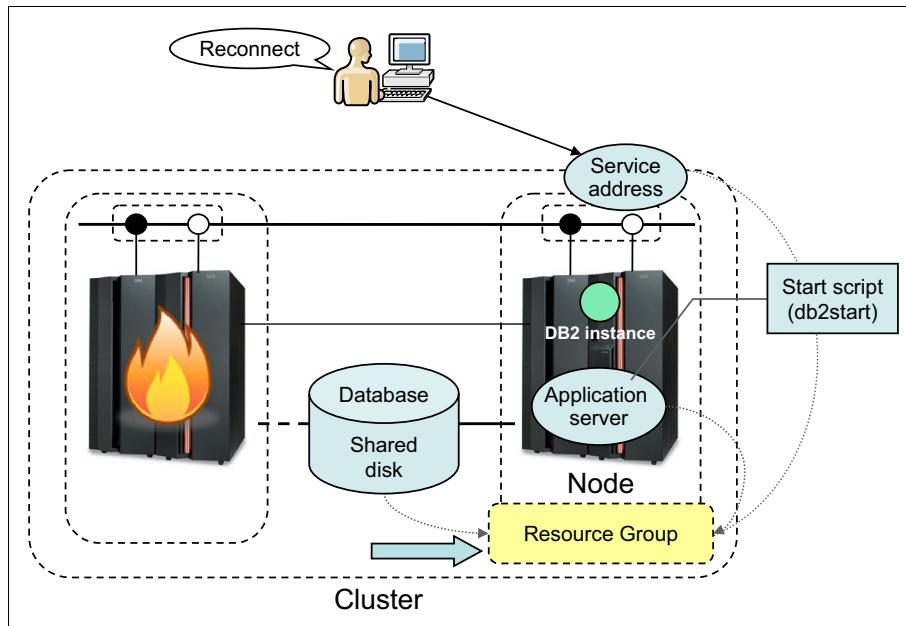


Figure 3-2 Failover (unplanned takeover)

► *Switch over* (planned takeover for maintenance)

You can switch over the resource group intentionally to the standby node for machine maintenance. This action is done by performing two PowerHA management operations:

- Stop PowerHA in takeover mode on the service node.
The application server stop script on the node that must release the resource group is run. The stop script gracefully stops all the applications and processes that access the shared disk.
- Move the resource group from the service node to the standby node.
The PowerHA process is still running on both nodes. PowerHA releases all resources, including shared disks and any service IP address that is related to the resource group. After it is released by the service node, the resource group is acquired on the standby node's side.

Figure 3-3 illustrates how the resource group moves during the switchover.

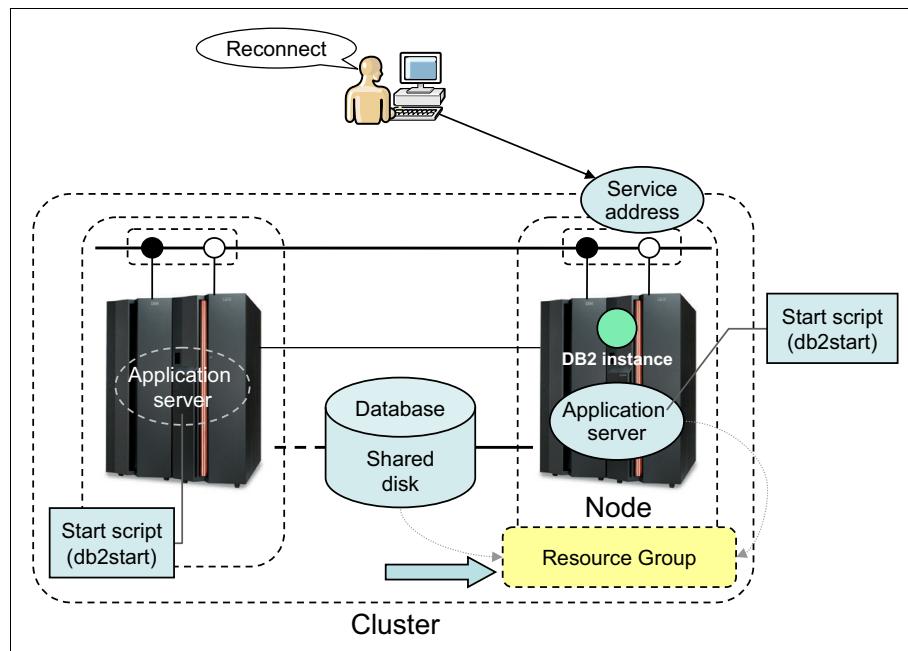


Figure 3-3 Switchover (planned takeover for maintenance)

3.3 Planning the PowerHA cluster

Planning a PowerHA cluster is similar to planning a high availability cluster with IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP). For more information, see 2.3, “Planning the high availability cluster” on page 27.

3.4 Setting up the PowerHA cluster

Setting up PowerHA usually requires the following steps:

1. Define the cluster and add nodes to the cluster.
2. Configure the PowerHA network and decide which method is 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 useful to replace the start and stop script entry with a test script (dummy script) to test the basic functions of PowerHA. After verifying that the PowerHA environment functions correctly, modify the start and stop script to work.

6. Configure the High Availability (HA) resource group.

This step groups the HA service IP, shared volume groups and file systems, the application server, and all the resources that are required by the application into one resource group. If there is a node failure, the PowerHA moves the resource group that is running on the failed node to the surviving node in the cluster.

7. Verify and synchronize the PowerHA configuration.

After you configure PowerHA, you must verify and synchronize the topology and resource group using the facility that is provided by PowerHA (for example, through SMIT command panels). Every time the PowerHA configuration is changed, you must resynchronize the cluster from the node where the change is made.

After the verification and synchronization is successful, you can start the PowerHA service.

3.4.1 PowerHA cluster setup planning

Before you set up an PowerHA environment, you must plan the cluster environment. Consider the following list of items:

- ▶ Physical nodes
- ▶ Network
- ▶ Application back-end/services software (for example, DB2)
- ▶ PowerHA configuration
- ▶ Application server start/stop scripts

Next, 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 that are provided in 3.4.2, “PowerHA configuration” on page 79.

Sample environment

Figure 3-4 shows the sample configuration of an active/standby PowerHA cluster configuration, with a normal DB2 instance in a shared disk configuration.

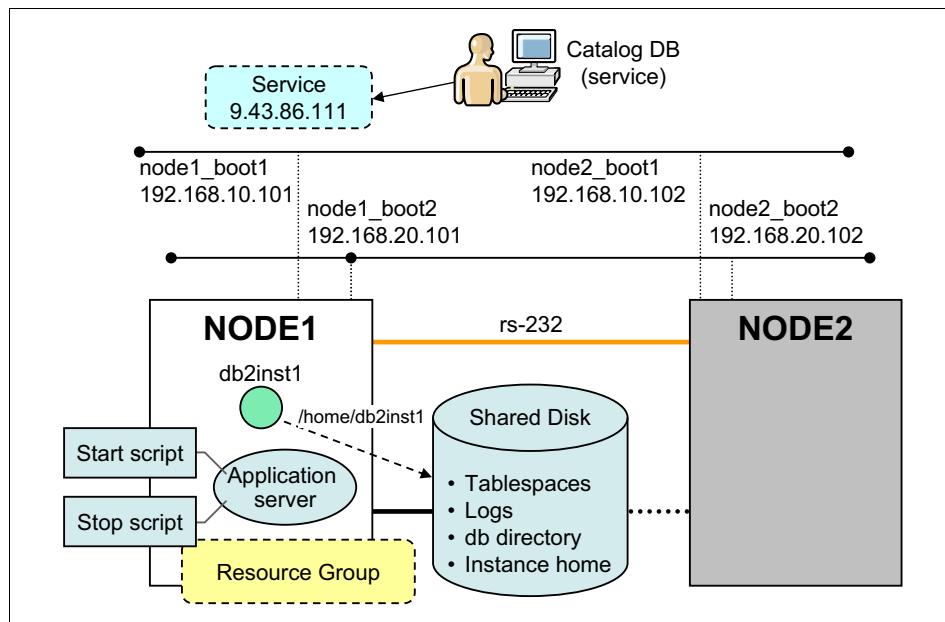


Figure 3-4 PowerHA disk shared cluster

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*, which normally provides 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 interfaces are both under the control of PowerHA. The service address for clients is added on one of these network interfaces.
- One Serial (RS-232C) network is configured for PowerHA keepalive. Use a serial network (non-TCP/IP network), because it makes PowerHA 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 all must be included in a single resource group.
- ▶ PowerHA 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* starts the database instance and restarts the database. The *application server stop script* stops the instance.

3.4.2 PowerHA configuration

The last section briefly described the planning stage before PowerHA implementation. This section introduces an outline of actually setting up PowerHA in a shared disk cluster configuration.

Checking the network connection

Before you configure PowerHA, the network must be configured properly. To check the network configurations for the cluster, complete the following steps:

1. Check that the IP addresses are configured on network interfaces on both nodes.
2. Check that the /etc/hosts file has all the entries of the IP addresses and their labels are the ones that are used by PowerHA.
3. Verify that the name resolution is working well by running **host**. If something is wrong, check and modify the /etc/hosts file.
4. Check the serial network connection. Use the subsidiary network for PowerHA keepalive to make PowerHA failure detection more secure. For more information, see *Administrating PowerHA SystemMirror*, found at:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/index.jsp?topic=%2Fcom.ibm.aix.powerha.admngd%2Fha_admin_kickoff.htm

Configuring the shared disk

This requirement is specific to an PowerHA shared disk configuration. Because shared disks are an important part of the PowerHA setup, this section lists the high-level steps that are needed to set up shared disk resources.

Here are some of the terms for AIX Logical Volume Manager, which are used in this section:

- ▶ Volume group (VG)
- ▶ Logical volume (LV)
- ▶ Journalized file system (JFS or JFS2)
- ▶ File system (FS)
- ▶ Log that maintains a consistent JFS (JFSLog or JFS2LOG)

To set up shared disk drives and the logical volume manager, complete the following steps:

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 by running **1vlstmajor**:

```
root@node1:/# 1vlstmajor  
58...
```

To add a volume group, you can use the **smit** menu:

```
#smitty vg
```

Select 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 3-1.

Example 3-1 Add volume group

Add an Original Volume Group

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

Fields]

VOLUME GROUP name

[Entry

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 VG by running 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 you create the first file system on a new VG, AIX automatically creates 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 by running **mk1v** before you create the first file system.

To create a JFS2 log with the name db2vgjfslog in the db2vg VG, run the following command:

```
# mk1v -t jfs2log -y db2vgjfslog db2vg 1
```

To format the JFS2 Log, run the following command, and select y when asked whether to destroy the LV:

```
# logform /dev/db2vgjfslog
logform: destroy /dev/db2vgjfslog (y)? y
```

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, run the following command:

```
# smit lv
```

In the Add a Logical Volume menu, select a VG name and press Enter. The Add a Logical Volume menu opens. Complete the fields (Example 3-2).

Example 3-2 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	[homelv01]
* 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 is created, create a file system that is associated with this LV. In this example, we create a JFS2 file system. To add a JFS2 file system on the previously defined LV, run the following command:

```
# smitty jfs2
```

Select Add an Enhanced Journaled File System on a Previously Defined Logical Volume and complete the fields (Example 3-3).

Example 3-3 Adding a JFS2 file system

Add an Enhanced Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[Entry
Fields]
* LOGICAL VOLUME name homelv01
+
* 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 by running the following command:

```
#mount /home/db2inst1
```

5. Unmount all of the FSs and deactivate the VG by running 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 you ever need to NFS export the file system, the major number for the VG must be identical on both nodes. Ensure that the VG is defined not to be activated automatically on reboot because it can be activated and controlled by PowerHA.

To import a VG, run the following command:

```
# smit vg
```

Select Import a Volume Group and complete the fields (Example 3-4).

Example 3-4 Importing a volume group

Import a Volume Group

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

Fields	[Entry]
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 by running the following command:

```
# 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 you create a DB2 instance user on node1. To accomplish this task, run the following commands:

- # 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, you can create the DB2 instance and database on the shared disk. On node1, complete the following steps:

1. Create the group for the DB2 instance.
Run **mkgroup** to create the group.
2. Create the user for the DB2 instance.
Run **mkuser** and **passwd**.
3. Change the ownership of the FSs and LVs.
4. Install DB2 and set up the license key. The DB2 product must 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 the 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 /home/db2inst1/sql1ib directory so that you can create the instance and control files on node 2, then repeat steps 1 - 5. Using this procedure allows DB2 to automatically perform steps such as adding service entries and modifying the db2nodes.cfg file for you. Then, recatalog the database by running the following command:

```
db2 catalog database dbname on <DBPATH>
```

Instead of deleting /home/db2inst1/sql1ib, you can repeat steps 1 - 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 file 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. This section explains what must be included in these application server scripts.

In the start scripts, 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 sql1ib 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 3.5, “Considerations for db2nodes.cfg file” on page 90.
- ▶ 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, you must stop all applications and processes that access the shared disk to ensure that the node can release the shared disk successfully. In DB2 terms, this means you must run **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 to get applications disconnected in a reasonable period.

A sample script file is packaged with DB2 ESE for AIX to help configure PowerHA 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 in the sql1ib/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 called by running **rc.db2pe db2inst1 start** and **rc.db2pe db2inst1 stop**.

Parallel System Support Programs (PSSP): rc.db2pe.eee is based on 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 PowerHA events, see *Data Recovery and High Availability Guide and Reference*, SC27-3870-00. This topic is also available in the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0007500.html>

Configuring PowerHA

smitty for AIX provides PowerHA configuration interfaces. The main steps are:

1. Add nodes to the PowerHA 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 the serial network and serial network device.
7. Verify and synchronize PowerHA configurations.
8. Basic verification of PowerHA configuration.

Here we expand upon each step:

1. Add nodes to PowerHA cluster by running the following command

```
#smitty sysmirror
```

You can run **smitty hacmp** as well.

From the PowerHA SystemMirror menu, select Cluster Nodes and Networks → Manage Nodes → Add a Node.

Enter the node name.

2. Add a service IP address by running the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resources → 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, run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resources → Configure User Applications (Scripts and Monitors) → Application Controller Scripts → Add Application Controller Scripts.

In the Add Application Controller Scripts menu, enter the controller name, and the complete paths of the start and stop scripts.

Start and stop scripts: The start and stop scripts that are called from the application controller must exist in the local directory on both nodes and have the same name.

4. Add a resource group.

To add a resource group, run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → 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 by running the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resource Groups → Change>Show Resources and Attributes for a Resource Group.

In the Change>Show Resources for a Resource Group menu (Example 3-5), enter the service IP label/address, application server name, volume groups, and file system information.

The resource group policies that we set in this example are shown in Example 3-5. These settings correspond to the *Rotating resource group* in former PowerHA versions, which means there is no priority for resource groups between nodes. For more information, see *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Planning PowerHA SystemMirror*, found at:

http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.powerha.plangd/hacmpplangd_pdf.pdf

Example 3-5 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.

Fields	[Entry]
Resource Group Name	db2_rg
Participating Nodes (Default Node Priority)	node1_bt
node2_bt	
Startup Policy	Online Using
Distribution Policy	Fallover To
Fallover Policy	
Next Priority Node In The List	

<pre> Fallback Policy Fallback> Service IP Labels/Addresses + Application Servers + Volume Groups + Use forced varyon of volume groups, if necessary Automatically Import Volume Groups Filesystems (empty is ALL for VGs specified)] </pre>	<pre> Never [service] [db2_server] [db2vg] false false + [/home/db2inst1] </pre> <hr/>
---	---

6. Define the serial network and serial network device.

If you have a serial network as the subsidiary keepalive network, you can configure this network from the **smit** menu.

7. Verify and synchronize PowerHA configurations.

After you define the PowerHA configuration from one node, you must verify and synchronize the cluster topology to the other node. Run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Nodes and Networks → Verify and Synchronize Cluster Configuration.

The PowerHA verification utility checks that the cluster definitions are the same on all nodes and provides diagnostic messages if errors are found.

The PowerHA configuration details can be found in the following publications:

- ▶ *PowerHA SystemMirror for AIX* PDF manual links:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/index.jsp?topic=%2Fcom.ibm.aix.powerha.navigation%2Fpowerha_pdf.htm
- ▶ *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Administering PowerHA SystemMirror*, found at:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.powerha.admngd/hacmpadmngd_pdf.pdf

- ▶ *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Planning PowerHA SystemMirror*, found at:
http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.powerha.plangd/hacmpplangd_pdf.pdf
- ▶ *Data Recovery and High Availability Guide and Reference*, SC27-3870-00

3.5 Considerations for db2nodes.cfg file

From Version 8 onwards, DB2 Enterprise Server Edition (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 must 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 must 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 does 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 3-6 shows a case where **db2start** fails when db2nodes.cfg does not match the host name.

Example 3-6 Error messages that are 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 are listed here:

1. Modify the file entry in the start script.
2. Run the **db2start** command with the **restart** option.
3. Run the **db2gcf** command with the **-u** option.
4. Use an alias in the hosts files.

Options 1, 2, and 3 modify the db2nodes.cfg file manually or automatically, while option 4 does not. Option 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 page also contains information about the format of the DB2 node configuration file for DB2 10.1:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.qb.server.doc/doc/r0006351.html>

3.5.1 Modifying the file entry in the start script

Perhaps the simplest of the listed methods, this method entails modifying the db2nodes.cfg file before you start 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 PowerHA Service node named node1 (node2 as standby), has a db2nodes.cfg file with the following content:

```
0 node1 0
```

If there is a takeover on node2, this db2nodes.cfg file is invalid. Therefore, the application start script can include a process to modify db2nodes.cfg to have the following content, 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 3-5 shows this concept graphically, with states shown before, during, and after the takeover by node 2 occurred.

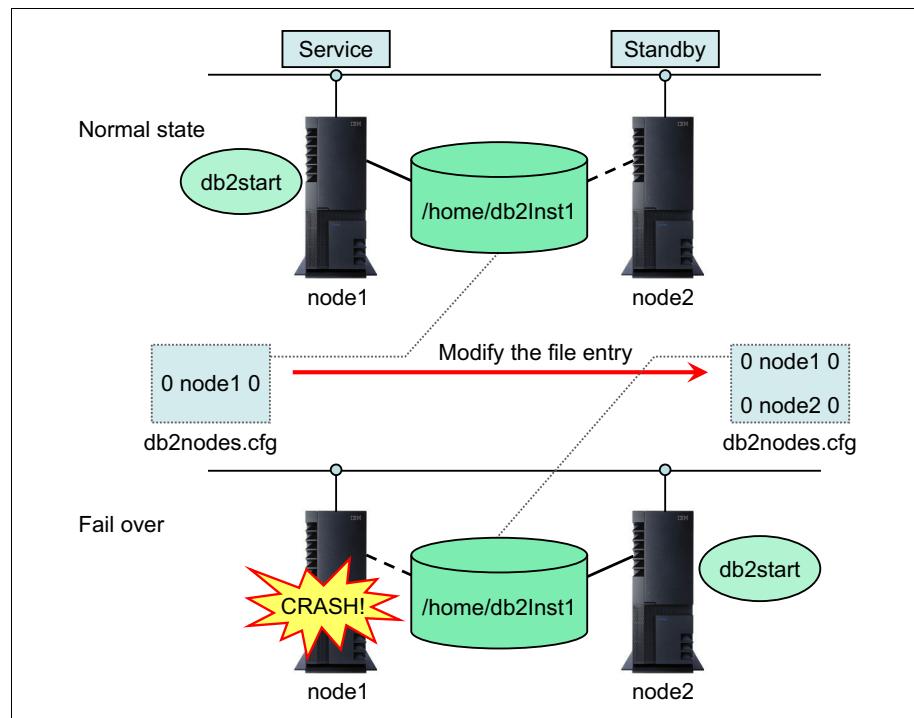


Figure 3-5 Modifying the `db2nodes.cfg` entry before starting DB2 on node2

3.5.2 Running 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 (Example 3-7). DB2 automatically modifies the `db2nodes.cfg` file with the host name you specified in the `restart` option.

Example 3-7 Restart option of the `db2start` command

```
db2start dbpartitionnum 0 restart hostname node2
```

Considerations

Because the **restart** option requires the permission of remote execution, you must configure remote shell (rsh) or Secure Shell (SSH) to be available in the cluster. This setting 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 because of the security risks that the remote shell might present.

- ▶ SSH configuration (available since DB2 8.2):

Install SSH, and set the public key and private key for the usage of the DB2 instance. Set the full path of the `ssh` command in the `DB2RSHCMD` registry value as follows:

```
db2set DB2RSHCMD=/usr/bin/ssh
```

3.5.3 Running the db2gcf command with the -u option

This option uses the `db2gcf` command. When you run 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 3-8 for a `db2gcf` command example and Example 3-9 for the `db2nodes.cfg` file that is modified by this command. You might notice in Example 3-9 that a fourth parameter is appended to `db2nodes.cfg`. This parameter is used for interconnect for Fast Communications Manager (FCM) communication.

Example 3-8 db2gcf command that is issued with the -u option

```
$ db2gcf -u -p 0 -i db2inst1
```

```
Instance : db2inst1
DB2 Start : Success
Partition 0 : Success
```

Example 3-9 Entry of db2nodes.cfg file that is modified by the db2gcf command

```
0 node2 0 node2
```

For more information about **db2gcf**, see *DB2 10.1 Command Reference*, SC27-3868-00, or the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0010986.html>

3.5.4 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 must 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 3-10.

Example 3-10 /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 3-11.

Example 3-11 /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 starts.

Next, the hosts entry in the /etc/netsvc.conf file must contain local as the first parameter. The hosts entry on both primary and standby systems then looks similar to Example 3-12.

Example 3-12 Entry of etc/netsvc.conf file

hosts=local,bind

By putting local as the first parameter, this forces the system to look in the /etc/hosts file for a host name entry before it goes 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 3-13, where db2host is the name that is defined in the hosts file on both systems.

Example 3-13 Entry of db2nodes.cfg file

```
0 db2host 0
```

After you complete these steps, failover and fallback testing should be done to ensure that the HA computing environment is working correctly.

3.6 Tuning tips for quick failover

When an outage on a primary database occurs, the steps that are 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, you must reduce the time that is taken in each step. This section describes the considerations for each step, which are shown in Figure 3-6.

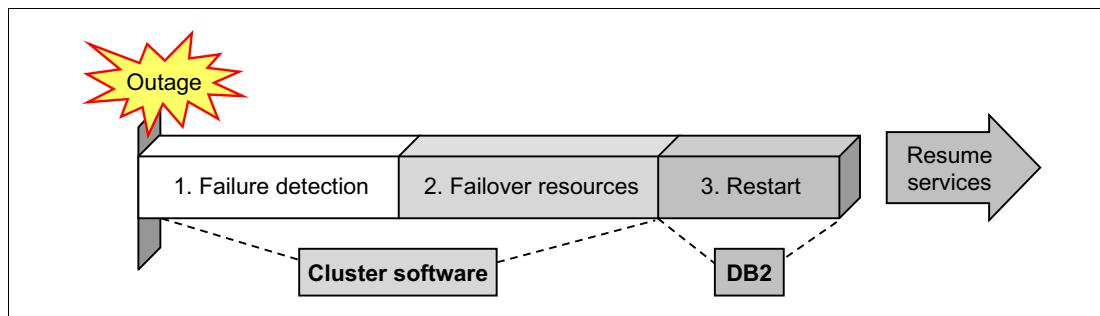


Figure 3-6 Time that is consumed by failover process

3.6.1 Failover of the resources

This section describes several options to speed up resource failover time.

IP address takeover

In previous versions of PowerHA, 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 *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Administering PowerHA SystemMirror*, the reduced time is because of fewer commands that are required when you move addresses.

For more information about IP Aliases in PowerHA, see the following PowerHA manuals:

- ▶ *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Administering PowerHA SystemMirror*, found at:
http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.powerha.admngd/hacmpadmngd_pdf.pdf
- ▶ *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Planning PowerHA SystemMirror*, found at:
http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.powerha.plangd/hacmpplangd_pdf.pdf

Disk resource failover

During a failover, PowerHA moves shared disks that are defined in the resource group from one server to another. This migration includes the following activities:

- ▶ *Varyon* (activate) VG By running **varyonvg**.
- ▶ 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 that are in the resource group, the longer it takes to failover. Therefore, the basic way to speed up disk resource takeover is to reduce the number of FSs in the resource group. This task can be accomplished by using raw device containers instead of FSs. With DB2 database managed space (DMS) on the raw device, you can decrease the disk resource failover time, because it is not necessary to mount and check consistency of FSs during failover.

You can also have a *concurrent access resource group* for the raw disk, reducing failover time even further. *Concurrent access volume group* with PowerHA 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 PowerHA requires the installation of an additional PowerHA file set.

For example:

```
cluster.es.clvm.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 situation means that they can be accessible from all nodes in the cluster simultaneously and there is a possibility of unintentional data corruption or loss. In this situation, software with data on a shared disk must ensure consistent data access, whether that be performed from within the software, or controlled externally through the application start/stop scripts.

Although DB2 databases do not tolerate concurrent access from multiple servers, in mixed/hybrid PowerHA/HADR mode, as described in Chapter 7, “HADR with clustering software” on page 223, DB2 databases use a shared disk on the primary cluster pair, and HADR writes to the standby on a third server node. Only one DB2 instance can actively use this shared disk at once.

To improve performance of failover/takeover, use a raw device for data storage. 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 of the shared disks, so using a DMS raw device for the table space container can improve failover performance.

- ▶ Catalog table space

You can specify the characteristics of the DB2 system catalog table space when you create databases by running **CREATE DATABASE**. Because the DB2 SYSCATSPACE is small in most cases, it is unlikely to contribute to reduction of failover time.

- ▶ Other database components

Other database components, such as database directory files, instance home and subdirectories, and archive log directories, must be created on FSs. The archive logs also can use Tivoli Storage Manager and avoid the issue altogether.

Because you still have FSs for some database components, PowerHA fast disk takeover can help fail over these file system resources. This feature enables faster resource group takeover using AIX Enhanced Concurrent Volume Groups (ECVG). 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 PowerHA software.

For all shared VGs created in enhanced concurrent mode that contain FSs, PowerHA activates the fast disk takeover feature. When PowerHA starts, all nodes in a resource group that share enhanced volume group varyons this VG in passive mode. When the resource group is brought online, the node that acquires the resources varyons the VG in active mode. This action reduces the time to varyon VGs, even though it still requires time to mount FSs.

For more information about fast disk takeover, see *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Planning PowerHA SystemMirror*, found at:

http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.powerha.plangd/hacmpplangd_pdf.pdf

There is another PowerHA option to speed up mounting FSs: parallel mount of FSs. You can use this feature to specify 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 ensure that all the errors in the FSs can be fixed if the node fails in the middle of file system I/O.

Starting and activating the database

After the resources are taken over by the standby node, re-enable the DB2 database for client use, which includes the following activities:

- ▶ Restart the DB2 instance (run **db2start**).
- ▶ Activate the database, including buffer pool activation and crash recovery.

If crash recovery is required, it might be 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 task 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 situation means that the time that is 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:

- ▶ Records to be rolled forward: The amount of log records that are committed, but are not written from the buffer pool to the disk.
- ▶ Records to be rolled back: The amount of log records that are not committed, but are written to disk.

To make crash recovery fast, you must 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. Here are some relevant tuning parameters:

- **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 the processes that are activated is determined by the **NUM_IOCLEANERS** database configuration parameter. 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 reduces the crash recovery time. Increase this parameter based on the number of the processor cores on the server.

- **SFTMAX** and **CHNGPGS_THRESH**:

The page cleaners are started according to the settings of the **SFTMAX** and **CHNGPGS_THRESH** database configuration parameters. Tuning these parameters lower than the default values reduces the crash recovery time, with the trade-off of increasing the load on certain system resources.

When the **SFTMAX** parameter is set lower, it can 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 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 setting triggers the asynchronous page cleaners to start cleaning the buffer pool more often. Committed data is written to the disk more often and less recovery time is needed in case of crash recovery.

Theoretically, 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 might 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.

For even further improved and proactive page cleaner activity, consider using the **DB2_USE_ALTERNATE_PAGE_CLEANING** registry variable. This registry variable makes **CHNGPGS_THRESH** redundant, as it no longer controls page cleaning.

- ▶ Records to be rolled back

It is important to frequently issue commits from the applications. This matter 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 coded online transaction performs most of its work without even holding locks on data. It needs only to gain locks momentarily before immediately committing to process requests after the user confirms an 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 roll back. These uses are often the same ones who complain when told that the answer is to commit as frequently as possible, meaning that they must code retry and restart logic into their batch jobs, or pursue alternatives such as unlogged loads with **copy yes** options.



DB2 with Microsoft Windows Failover Cluster

This chapter explains a sample setup of a highly available DB2 environment with Microsoft Windows Server 2008 R2 Failover Cluster. It explains the fundamental concepts of Microsoft Failover Cluster to help you understand its architecture and the basic configuration procedures.

This chapter describes how to configure a DB2 instance and other cluster resources to become the highly available resources in the Failover Cluster. It demonstrates the configuration steps using both the automatic utility **db2mscs** and the manual steps. It also demonstrates a sample test scenario using IBM Data Studio, where you can see the cluster in action.

Finally, this chapter explains the steps that you must follow to upgrade your servers with minimal downtime.

This chapter covers the following topics:

- ▶ Failover Cluster concepts
- ▶ Minimal steps to cluster a DB2 instance
- ▶ Creating a server cluster
- ▶ Installing DB2
- ▶ Creating a DB2 instance
- ▶ Manually configuring a DB2 instance
- ▶ Using db2mscs to configure a DB2 instance

- ▶ Testing a cluster
- ▶ Upgrading your instance

4.1 Failover Cluster concepts

Microsoft Windows Server 2008 provides two techniques for clustering: Windows Server Failover Cluster and Network Load Balancing (NLB). Failover Cluster helps setting up highly available environments for stateful applications. NLB provides scalability and increases availability for stateless applications (such as proxy or web servers). In this chapter, you are concerned only with Failover Clusters.

Failover Cluster availability: The Failover Clustering feature is included in Windows Server 2008 Enterprise and Windows Server 2008 Datacenter. It is not included in Windows Server 2008 Standard or Windows Web Server 2008.

This chapter is not intended to give a detailed explanation of Microsoft Failover Cluster, but rather to introduce its fundamental concepts so that you can consider using this feature with DB2 when you select a clustering product for a DB2 application.

This section introduces some concepts that are specific to Failover Cluster. You might want to review some of general high availability concepts that were introduced in Chapter 1, “DB2 high availability and disaster recovery overview” on page 1.

4.1.1 Failover Cluster overview

Windows Server 2008 and Windows Server 2008 R2 Failover Cluster (formerly known as Microsoft Server Clusters) introduce some new features and enhancements. The new improvements include easier cluster setup and management, enhanced security and stability, improved networking, and how the cluster communicates with the storage. The cluster can now support up to 16 nodes instead of eight nodes.¹

¹ For 32-bit operating system, you still have only eight nodes.

New quorum model

For a cluster that supports up to 16 nodes, the shared quorum model can be a single point of failure. To eliminate this single point of failure, the quorum model was changed to a majority model. A cluster can continue running if the majority of nodes can communicate to each other. Every node on the cluster has a vote and in some cases a disk or a file share can also vote. The voting majority continues running as a cluster. The minority stops running as a cluster. Quorum is about controlling who you want to have a vote. Windows Server 2008 Failover Cluster supports four quorum modes:

- ▶ **Node Majority**

Each node that is available and in communication can vote. The cluster functions only with a majority of the votes, that is, more than half. This mode is best suited for clusters with an odd number of nodes.

- ▶ **Node and Disk Majority**

All nodes and the witness disk that are available and in communication can vote. The cluster functions only with a majority of the votes, that is, more than half. This mode is best suited for clusters with an even number of nodes.

- ▶ **Node and File Share Majority**

The same as Node and Disk Majority, except for using a file share instead of a shared disk.

- ▶ **No Majority: Disk Only**

The cluster has quorum if one node is available and in communication with a specific disk in the cluster storage. This mode is the same as the previous shared quorum disk in Microsoft Windows Server 2003. For more information, see:

<http://technet.microsoft.com/en-us/library/cc731739.aspx>

This new quorum model introduced some new terminology: *disk witness*, *file share witness*, and *vote*. The witness concept represents a dedicated resource for use by the cluster as an arbitration point when nodes lose communication with each other. It is a new name that represents the old quorum concept.

Here is an overview of these new terms:

Disk Witness

A dedicated small NTFS disk (not less than 512 MB) that is on the cluster group. It can vote to the quorum. It stores an updated version of the cluster database to maintain the cluster state and configurations independently of node failures. It helps avoid split-brain scenarios in the cluster.

File Share Witness	A file share available to all nodes in the cluster. It also can vote to the quorum. It does not store a version of the cluster database. It helps avoid split-brain scenarios.
Vote	Majority (quorum) is calculated based on votes. Cluster nodes, disk witness, or file share witness can have a vote that is based on your selected quorum mode.

4.1.2 Windows Failover Cluster definitions

This section introduces a few concepts that are necessary to understand how Failover Clusters work in Windows Server.

Resource type

In Failover Cluster, a resource type is a service, such as web service, IP address, or a device, such as a shared disk, that must be made highly available.

For every type of resource to be manipulated, you need a handler, which is an interface that manipulates this resource. For example, if what you want to make highly available is a shared disk, you need a mechanism to:

- ▶ Detect that the resource is working properly
- ▶ Detect the availability of the disk from each node
- ▶ Failover or fallback the resource among the nodes as necessary

In the case of Failover 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), Failover Cluster provides a default DLL that can handle them.

In the case of DB2, a new type of resource that is known as DB2 should be created. We show how to add the DB2 resource type to Server Cluster in 4.6.1, “Adding the DB2 resource type” on page 123.

Resource

A resource is any element that you want to make highly available in the system. Common resources that you can find are:

- ▶ IP addresses
- ▶ DB2 instances
- ▶ Shared disks

Groups

A cluster group is a set of resources that are interdependent with 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. After a group is successfully created, you can select the preferred owner node for this group. Failover and failback always act on groups.

4.1.3 Managing Failover Cluster

This section briefly describes how to install the Failover Cluster feature and what are the tools that you can use to manage this functionality.

Installing the Failover Cluster feature

As of Windows Server 2008, clustering software is not installed by default. It is included as a pluggable feature in the operating system. You must add the Failover Cluster feature to all servers that you want to include in the cluster. To add the Failover Clustering feature, complete the following steps:

1. Click **Start** → **Administrative Tools** → **Server Manager** and, under Features Summary, click **Add Features**.²
2. In the Add Features wizard, click **Failover Clustering** and then click **Install**.

Available tools to manage Failover Cluster

Failover Cluster can be managed by using a GUI or a command-line interface (CLI). The GUI interface is a snap-in for Microsoft Management Console (MMC). You can use the Failover Cluster Manager snap-in to validate failover cluster configurations, create and manage existing failover clusters, and migrate some resource settings to a cluster that is running Windows Server 2008 R2. For CLI management, run **Cluster**. Windows Server 2008 R2 introduces a new PowerShell Cmdlets feature for Failover Clusters. Windows PowerShell provides a scripting interface for actions that you might otherwise perform with the Failover Cluster Manager snap-in. For more information about this topic, see:

[http://technet.microsoft.com/en-us/library/ee619751\(v=ws.10\).aspx](http://technet.microsoft.com/en-us/library/ee619751(v=ws.10).aspx)

² You can also open the Add Features dialog box by clicking **Initial Configuration Tasks** and, under **Customize This Server**, click **Add features**.

4.2 Minimal steps to cluster a DB2 instance

To set up your cluster using Windows Server 2008 R2 Failover Cluster, you need, at minimum, the following resources:

- ▶ A Windows 2008 domain.
- ▶ Two machines to work as the nodes in the cluster using the Windows Server 2008 R2 editions that support the Failover Cluster.
- ▶ Two shared disks that can be accessed simultaneously from all the nodes.
- ▶ Two network cards in each node (not mandatory, but a good idea).

In our lab example, we set up VMWare images with the following resources:

- ▶ Two machines named winNode101 and winNode102, running Windows Server 2008 R2 Enterprise at the torolab.ibm.com domain.
- ▶ Each node has two network cards, one for the private network and the other for the public network.
- ▶ Three shared disks were created:
 - Drive Q: This disk was created to be the quorum of the cluster.
 - Drives E and S: These disks are used for DB2 table spaces and instance profile storage.

Here is the procedure to set up a highly available DB2 environment with Failover Cluster. We provide detailed steps for creating a Windows Server Failover Cluster and how to configure it with IBM DB2 in the next few sections.

1. Install a Windows Server 2008 R2 edition that supports Failover Clustering on each cluster node.
2. Configure the shared disks devices and ensure that each node has access to the shared resources.
3. Add your nodes to a Windows Domain.
4. Add the Failover Cluster feature to all nodes.
5. Use the Validate a Cluster wizard to validate your system configurations.
6. Create a cluster using the Failover Cluster Create a Cluster wizard.
7. Install DB2.
8. Create a DB2 instance.
9. Make the DB2 instance highly available.
10. Test your cluster.

4.3 Creating a server cluster

To create a cluster, a Windows Domain is required. The Failover Cluster creates the needed objects on the domain and assigns them specific permissions.

Before proceeding, assume that you completed the following steps:

1. Added your servers to a domain.
2. Added the Failover Cluster feature to all nodes, as described in “Installing the Failover Cluster feature” on page 105.
3. Properly configured and connected your networks and storage on all nodes.

After successfully completing these steps, you can proceed with:

- ▶ Validating your system compliance with Failover Cluster
- ▶ Creating the cluster in the domain

Subsequent sections explain the remaining steps to configure your DB2 instance to make it highly available.

4.3.1 Validating your system

Microsoft Failover Cluster provides a validation wizard from which you can validate your system. If your system does not pass the validation tests, your cluster is not supported by Microsoft. In addition, your hardware must be marked as “Certified for Windows Server 2008 R2”. Your two machines should have the same operating system version, batch level, hardware configurations, and settings. For more information about these requirements, see:

<http://technet.microsoft.com/en-us/library/cc771404.aspx>

To start validating your system, click **Start** → **Administrative Tools** → **Failover Cluster Manager**. The Failover Cluster Manager snap-in window opens (Figure 4-1).

Logging in: If you are not logged in to this machine using a domain user, you get a warning message, and the Failover Cluster Management tasks become unavailable.

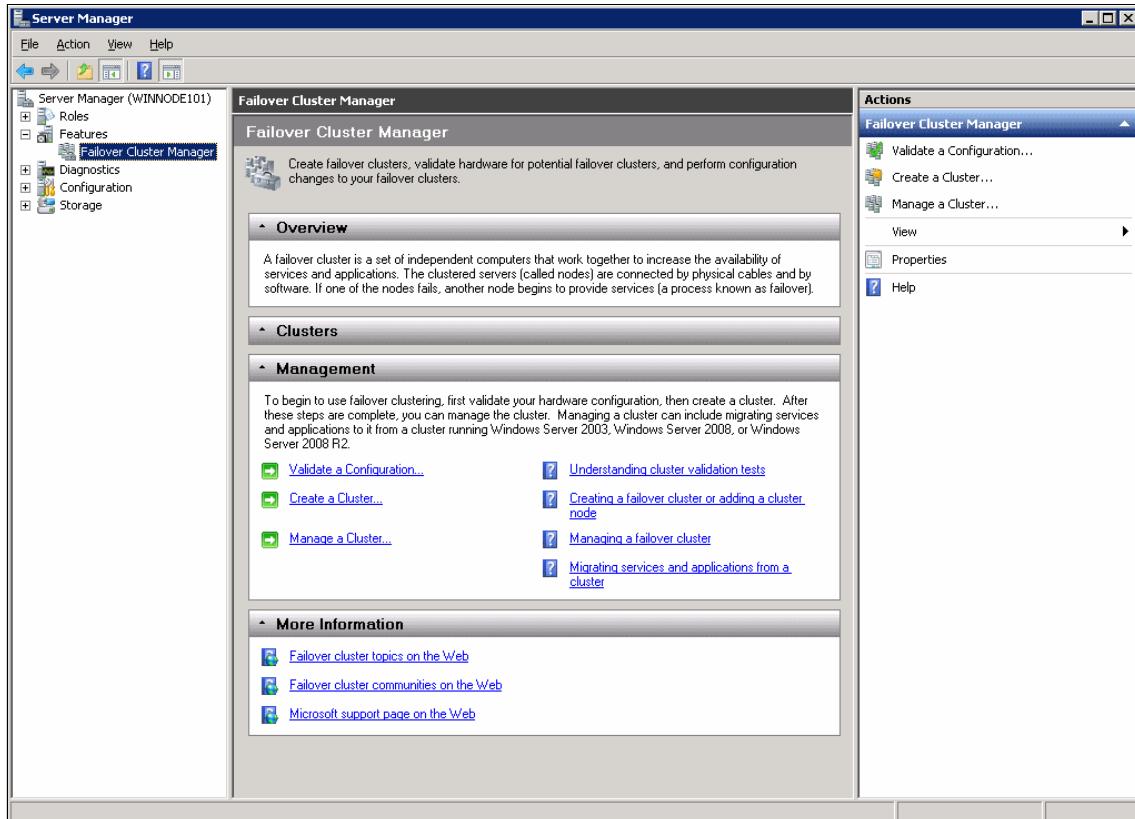


Figure 4-1 Failover Cluster Manager snap-in window

To validate your system configuration for Microsoft Failover Cluster, complete the following steps:

1. From the Failover Cluster Manager window (Figure 4-1), click **Validate a Configuration** in the Management section to start the validation wizard.

2. The Before You Begin window opens (Figure 4-2). This window displays some helpful information about the current wizard and provides some links to the Microsoft website. Click **Next**.

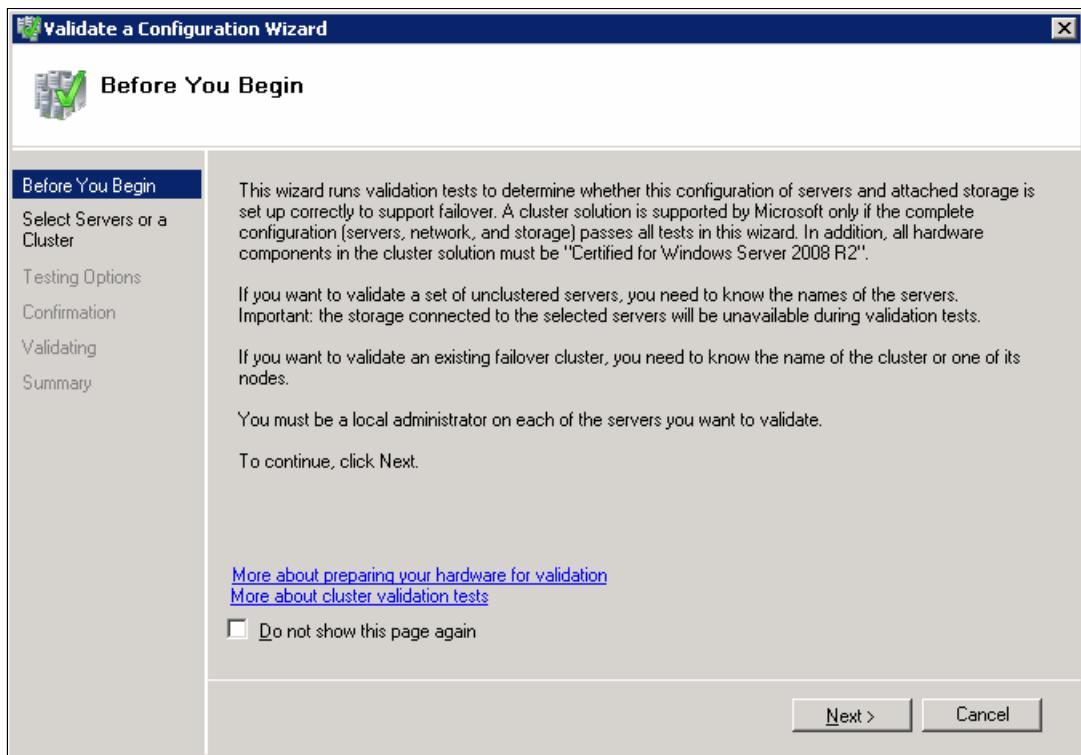


Figure 4-2 Validate a Configuration wizard

3. From the Select Servers window (Figure 4-3), enter the names of your servers to be validated and press the Enter key. You can add more than one server at once by separating them with semicolons (;). In this window, you cannot add a node that is already a member of another cluster. Your nodes can be a member of only one cluster. After you add your nodes, click **Next**.

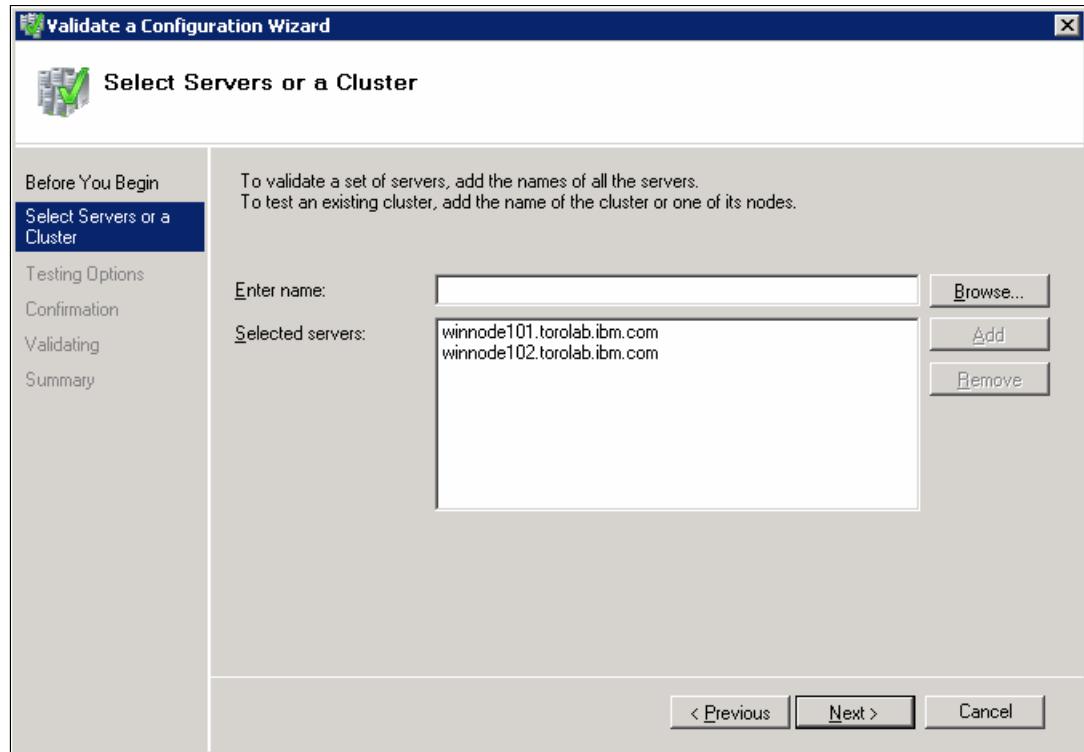


Figure 4-3 *Select Servers or a Cluster* window

4. The next window prompts you to choose the tests to run. You can choose to run all the tests or run only certain tests (Figure 4-4). We selected **Run all tests (recommended)**. Click **Next**.

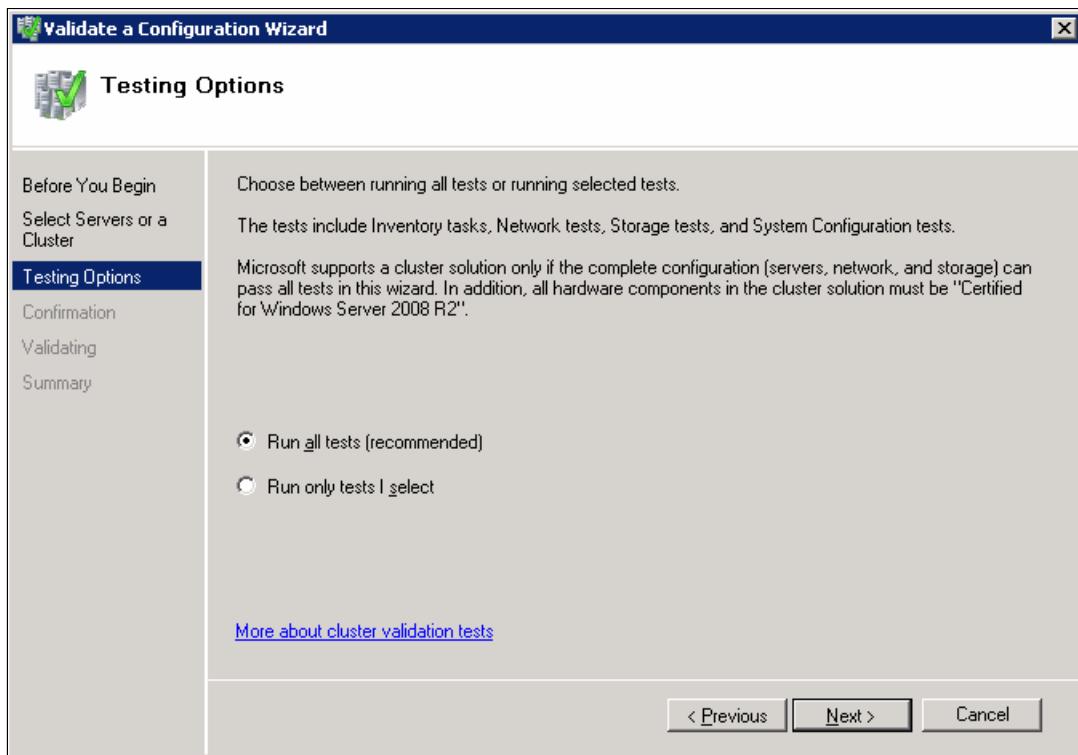


Figure 4-4 Select validations tests

5. A confirmation window opens and lists the selected tests (Figure 4-5). You can get back and add or remove some specific tests. After you are done, click **Next** to start the validation process.

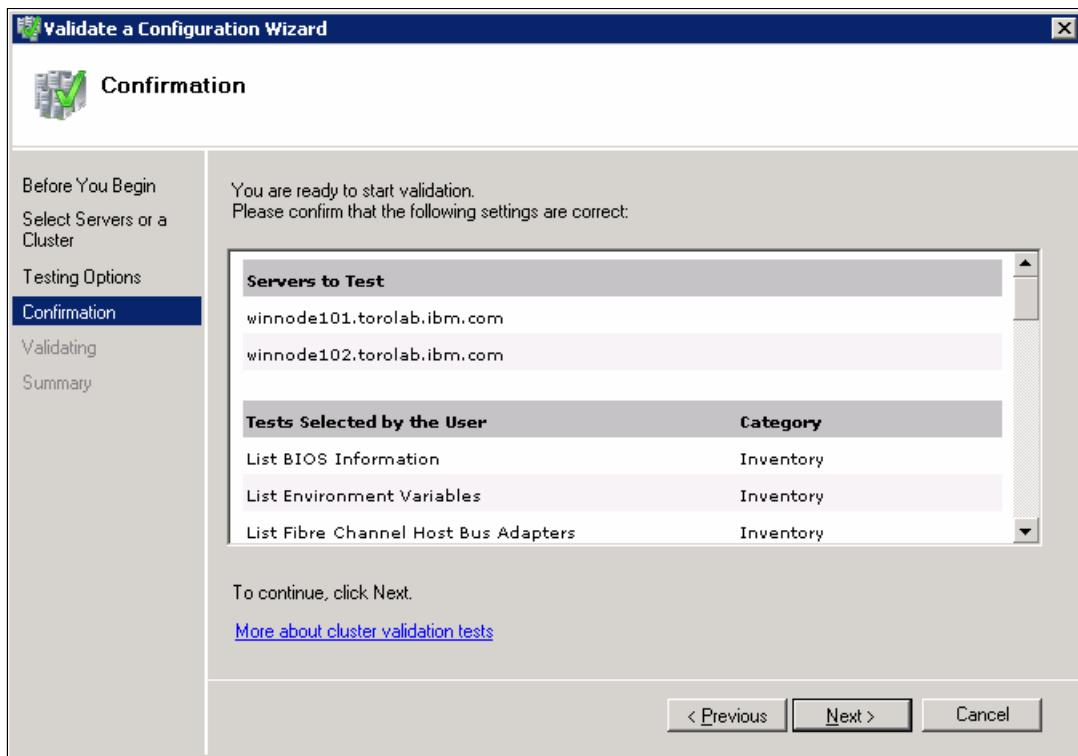


Figure 4-5 Selected validation tests confirmation

The validation tests are running (Figure 4-6). Wait for them to complete.

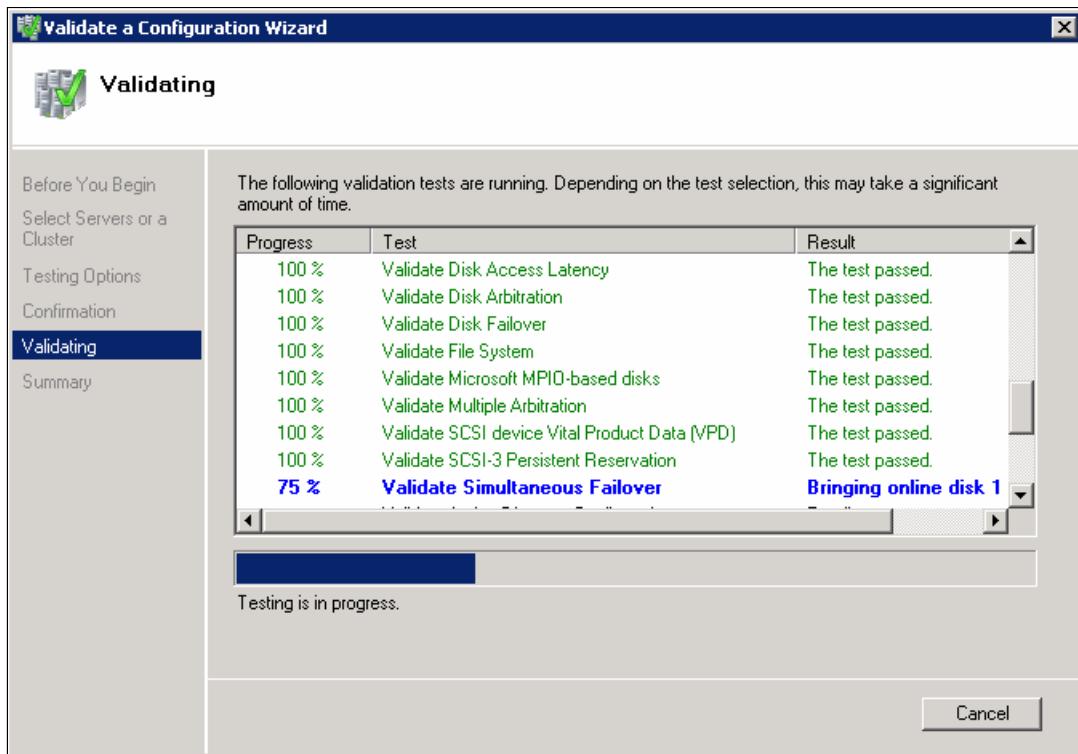


Figure 4-6 Validation tests that are running

- After successfully completing the system validation, a window opens with the summary report of the validation results (Figure 4-7). If you receive errors, try to correct these errors first. You might need to view the detailed report by clicking **View Report** to obtain more details about the failing tests.

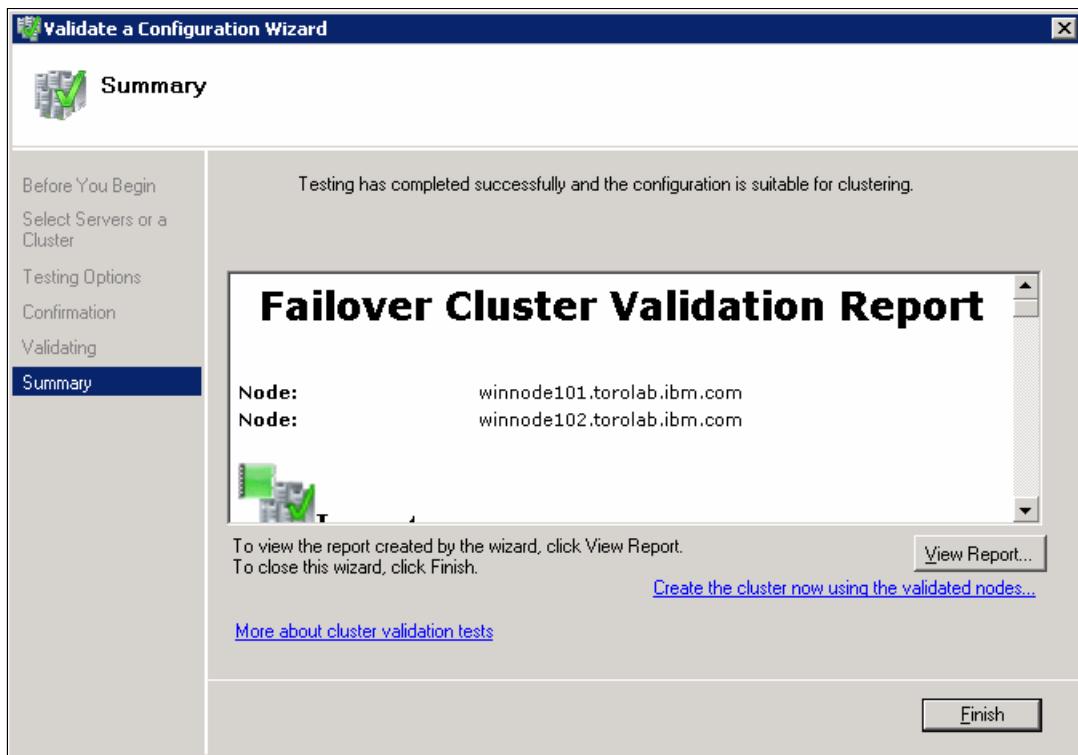


Figure 4-7 Validation summary report

Virtualized environments: If you are using a virtualized environment -and cloned your machines from each other or used the same template, you might get an error with the network configurations. For example, you might get “An item with the same key has already been added.” error.

What happened is that your network adapters have the same GUID on both nodes (because they are clones). The validation tools expect this Globally Unique Identifier to be unique. You can check the GUIDs by validating the output of this power shell command:

```
Get-WmiObject Win32_NetworkAdapter | fl Name,GUID
```

Compare the output on both nodes to verify GUID uniqueness. If the GUIDs match, delete the network adapter from one of the nodes and rescan for hardware changes to obtain the new network adapter that is installed with the new GUID.

Now you have completed the validation of your environment and are ready to create a cluster.

4.3.2 Creating a cluster in the domain

To create a cluster in the domain, complete the following steps. To create a cluster, a shared disk resource must exist. The supported types of storage are Serial Attached SCSI (SAS), iSCSI, and Fibre Channel.

1. Open Failover Cluster Manager, as described in 4.3.1, “Validating your system” on page 107

2. Under Features from the left navigation tree, right-click **Failover Cluster Management** and click **Create a Cluster**. The Create Cluster wizard introductory window opens (Figure 4-8). Click **Next**.

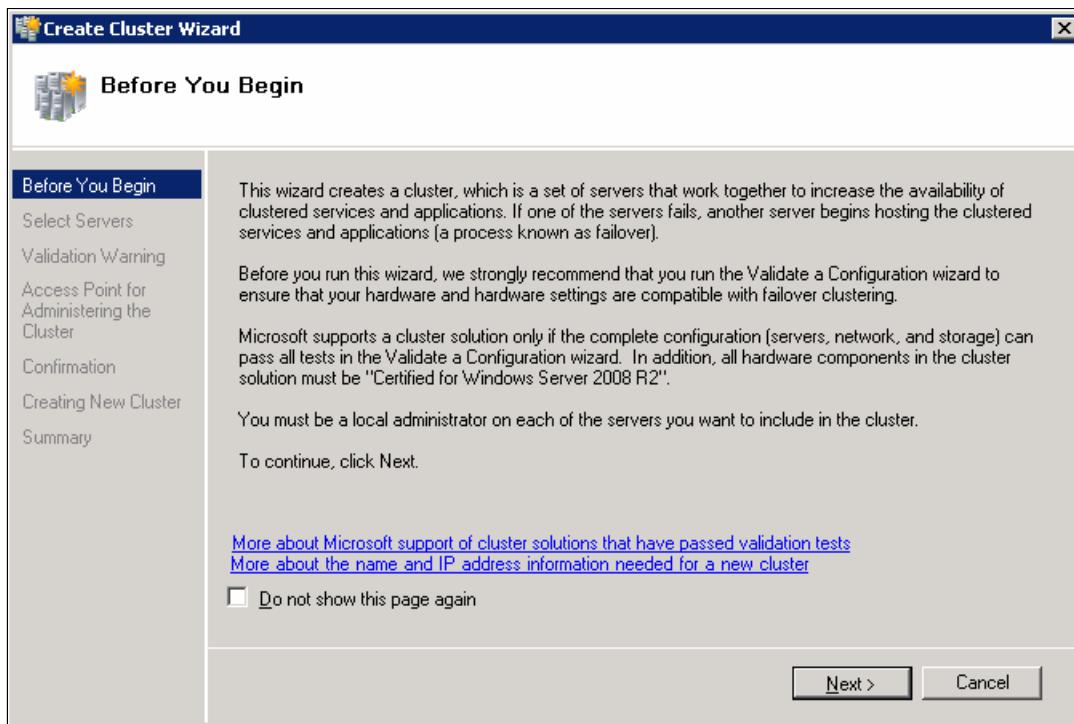


Figure 4-8 Create Cluster wizard

3. In the Select Servers window, add your cluster nodes as you did in Figure 4-3 on page 110. Select your validated servers and click **Next**.

4. If you have not run the Validate Cluster wizard before or your validation did not completely pass all the tests, you are prompted to run Cluster Validation. If your system passed all tests successfully, you do not get this dialog box. Instead, the Access Point for Administering the Cluster window opens (Figure 4-9). It asks you for the cluster name and access point (IP address) for administering the cluster. Input you cluster name (DB2Clust) and Access point IP (9.26.20.103) and click **Next**.

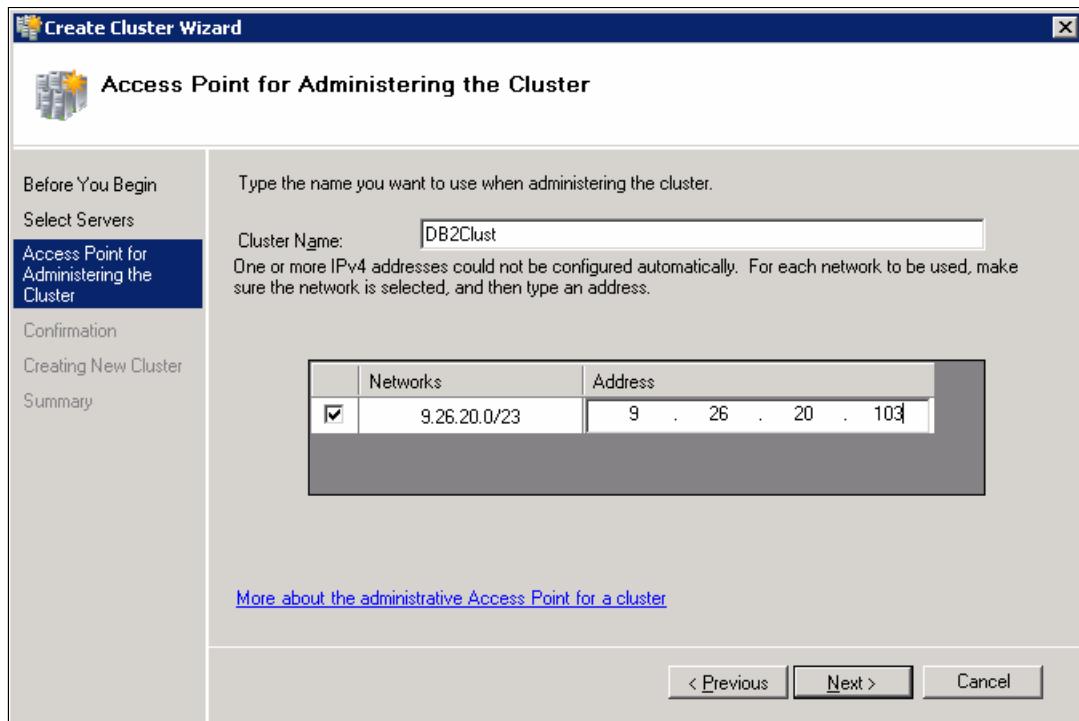


Figure 4-9 Access Point for Administering the Cluster

New computer object: After successful completion, Failover Cluster creates a computer object in the domain (DB2Clust) and assigns it an IP address (9.26.20.103). Remember this object, as you need to assign certain permissions to it later in this chapter. If you forgot this step, your cluster does not work properly.

5. A confirmation window opens with the selected configurations you provided (Figure 4-10). Click **Next** to start creating the cluster.

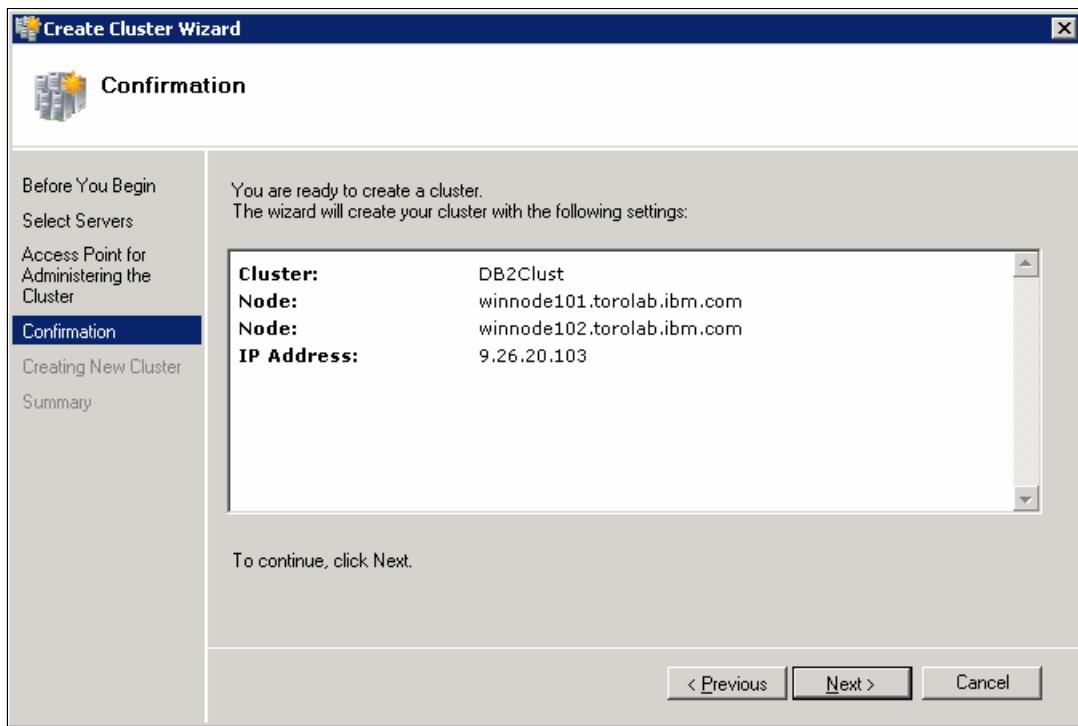


Figure 4-10 Cluster configurations confirmation

6. The Creating New Cluster window opens. It shows the creation process' progress (Figure 4-11).

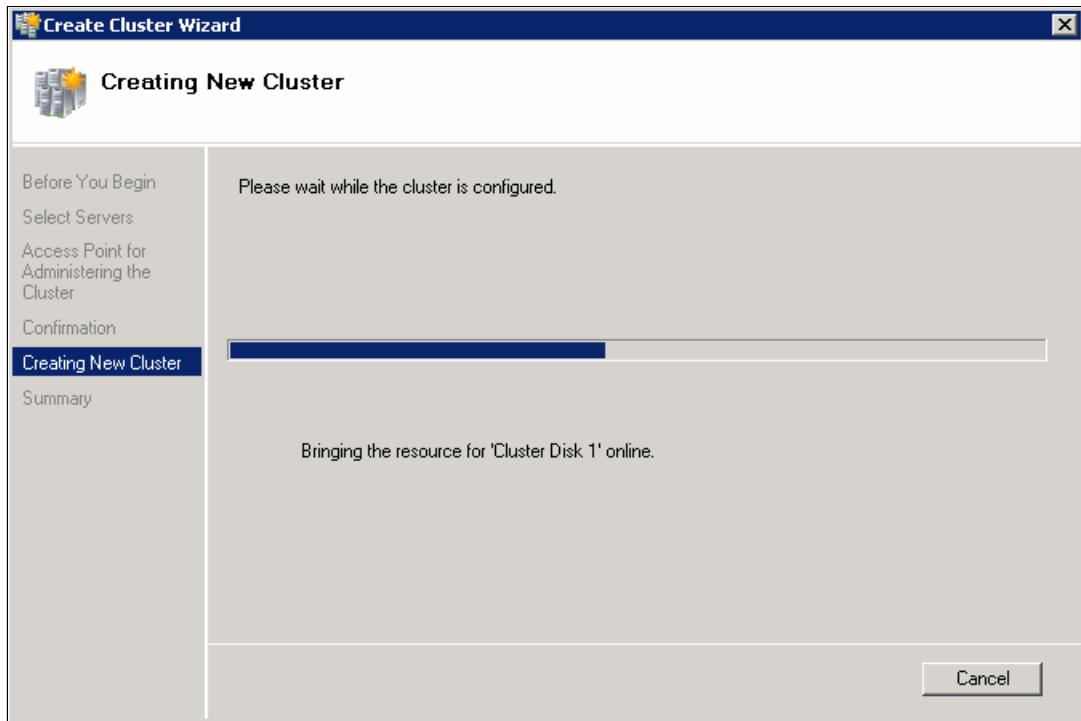


Figure 4-11 Creating cluster progress

7. After the cluster creation completes, a window opens and shows a summary report (Figure 4-12).

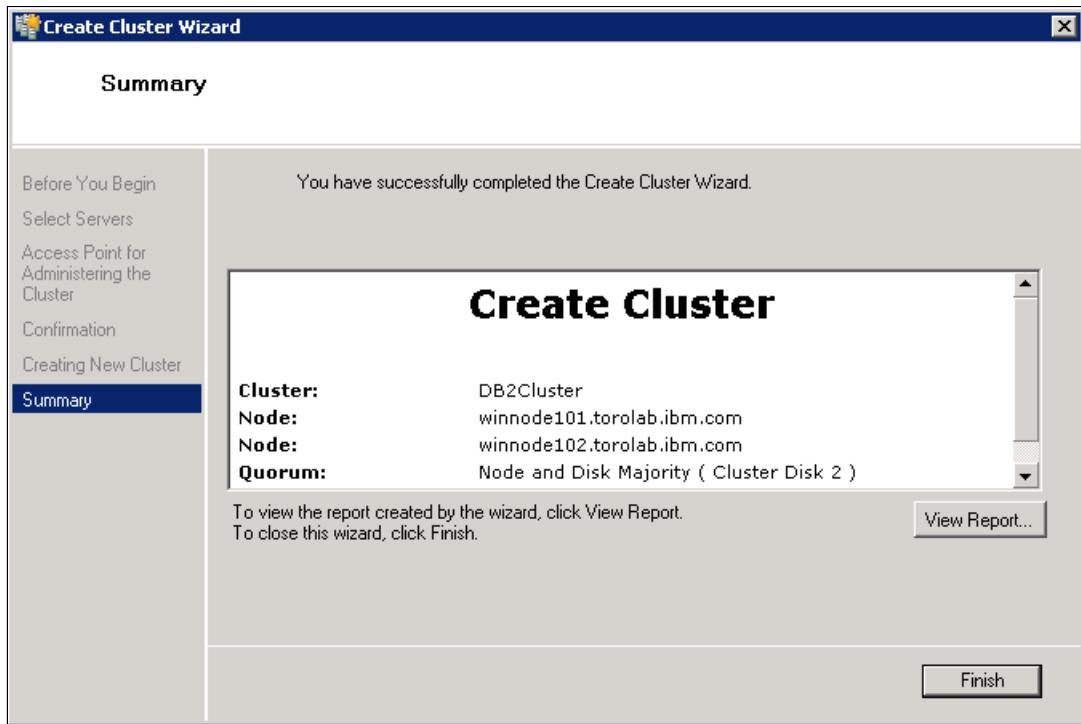


Figure 4-12 Create cluster summary

4.4 Installing DB2

DB2 should be installed on all the nodes in the cluster. For the DB2 installation procedure, see the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.qb.server.doc%2Fdoc%2Ft0008099.html>

While you are installing DB2, ensure that you enable extended security using domain groups. If you have DB2 installed, see 4.6.5, “Configuring security settings” on page 142 for the steps to correctly configure DB2 security for the cluster.

4.5 Creating a DB2 instance

DB2 creates a default instance during the DB2 installation on Windows. However, we create an instance to configure the DB2 cluster in this example.

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 create a DB2 instance in winNode102. The instance name is DB2HA and the owner of the instance is the domain user anas.mosaad.

Figure 4-13 shows the command that is used to create the instance and the result of the command. **db2icrt** requests the password of the domain user ID specified in the **-u** option.

The second **db2ilist** command in the same panel lists the instances in the local machine. Use the **-w** flag with 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.

```
C:\Administrator: DB2 CLP - DB2COPY1
C:\IBM\SQLLIB\BIN>db2ilist
DB2

C:\IBM\SQLLIB\BIN>db2icrt db2ha -s ESE -u torolab\anas.mosaad
Enter current password for torolab\anas.mosaad:
DB20000I The DB2ICRT command completed successfully.

C:\IBM\SQLLIB\BIN>db2ilist
DB2HA
DB2

C:\IBM\SQLLIB\BIN>
```

Figure 4-13 Create an instance DB2HA

If you open Windows Services (by clicking **Start** → **Administrative Tools** → **Services**), you should see a new 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 - DB2COPY1 - DB2HA-0.

In the case of a WSE instance, the name of the service has the suffix **<INSTANCE-NAME>**. If you created a WSE instance, the service is similar to DB2 - DB2COPY1 -DB2HA.

This name is important because it becomes the Windows Registry Key Name of the service. The following entries must exist in the Windows registry:

- ▶ ESE instance:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DB2HA-0

- ▶ WSE instance:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\DB2HA

The cluster software looks to this entry to start a highly available DB2 instance. This setup usually leads to the second most common problem in setting up DB2 with Windows Failover Cluster: If you name the DB2 resource with a name that is different from this entry, the resource does *not* find the DB2 server and does not start.

In the following sections, we describe two different methods to make the DB2 instance highly available with Windows Cluster: the manual method and the semi-automatic method using **db2mscs**. We show only a hot standby configuration for a single partition database here. For more information about mutual takeover and multiple partition configuration, see the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.ha.doc%2Fdoc%2Fc0007402.html>

4.6 Manually configuring a DB2 instance

Although there is a utility named **db2mscs** that can do most of the configuration of a DB2 instance, you should understand what steps are involved in making a DB2 instance highly available in a Failover Cluster environment so that you are able to diagnose any problem that occurs.

To configure a DB2 instance in a Failover Cluster to provide high availability, complete the following steps:

1. Add a DB2 resource type to the cluster.
2. Create cluster resources:
 - a. Create a Cluster Service or Application to group all the resources for the instance.
 - b. Create or add the supporting resources to the DB2 group:
 - i. A highly available IP address is mandatory.
 - ii. A highly available shared disk to store an instance profile is mandatory.

- iii. If this database is a multi-partitioned database, the instance profile directory must be shared and made highly available.
 - c. Add a DB2 Server resource to the cluster.
3. Link a DB2 Server resource with the DB2 instance.
 4. Add the instance to the remaining nodes.

4.6.1 Adding the DB2 resource type

After DB2 is installed, you must create a DB2 resource type in Failover Cluster. If you want Windows Server Failover Cluster to manage the DB2 instance, the DB2 instance must be a resource in the cluster.

If you click **Failover Cluster Management → Properties**, and then click **Resource Type** tab, you can view the current resource types that can be managed by the Failover Cluster (Figure 4-14). After you add a DB2 resource type, you can find it under the user-defined resource types.

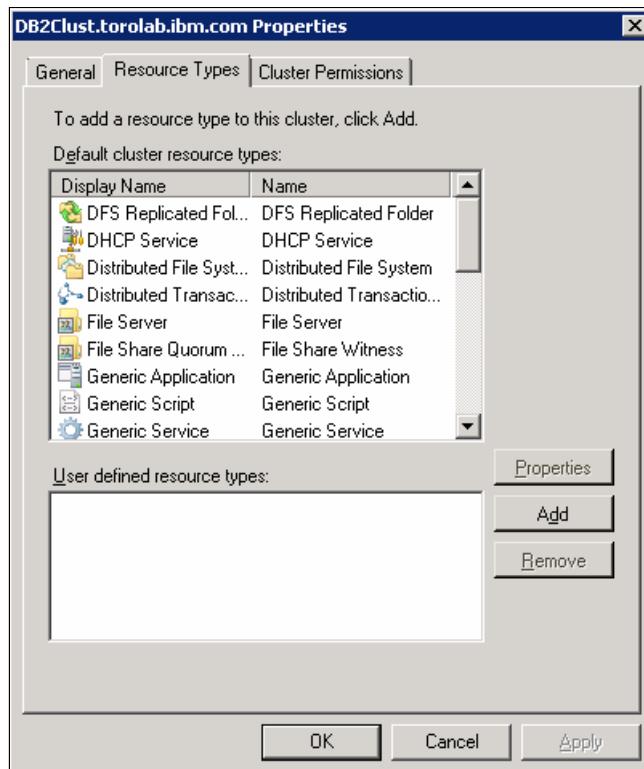


Figure 4-14 Cluster resource types

The DB2 resource type can be added to this list by running **db2wolfi**. This utility accepts one parameter:

```
db2wolfi u | i
```

Where:

- ▶ **u**: Unregister the DB2 resource type.
- ▶ **i**: Install the resource type.

After **db2wolfi** is run successfully, the resource type “DB2 Server” is added. You can view the cluster resource types from the command line by running **cluster** (Figure 4-15).

```
C:\ Administrator: DB2 CLP - DB2COPY1
C:\IBM\SQLLIB\BIN>db2wolfi i
ok
C:\IBM\SQLLIB\BIN>db2wolfi i
Error : 183
C:\IBM\SQLLIB\BIN>_
C:\IBM\SQLLIB\BIN>cluster restype
Listing all available resource types:

Display Name          Resource Type Name
-----              -----
DB2 Server           DB2 Server
DFS Replicated Folder DFS Replicated Folder
DHCP Service         DHCP Service
Distributed File System Distributed File System
Distributed Transaction Coordinator Distributed Transaction Coordinator
File Server          File Server
File Share Quorum Witness File Share Witness
Generic Application Generic Application
Generic Script       Generic Script
Generic Service      Generic Service
IP Address           IP Address
IPv6 Address         IPv6 Address
IPv6 Tunnel Address  IPv6 Tunnel Address
<Resource Type Unavailable> MSMQ
<Resource Type Unavailable> MSMQTriggers
iSNS Cluster Resource Microsoft iSNS
NFS Share             NFS Share
Network Name          Network Name
Physical Disk         Physical Disk
Print Spooler          Print Spooler
Virtual Machine       Virtual Machine
Virtual Machine Configuration Virtual Machine Configuration
Volume Shadow Copy Service Task Volume Shadow Copy Service Task
WINS Service          WINS Service

C:\IBM\SQLLIB\BIN>_
```

Figure 4-15 DB2Wolfi utility execution - adding the DB2 Server resource type

If you click **DB2Clust** → **Properties** → **Resource Types**, you can find the same objects that you viewed by running **cluster res-type** (Figure 4-16).

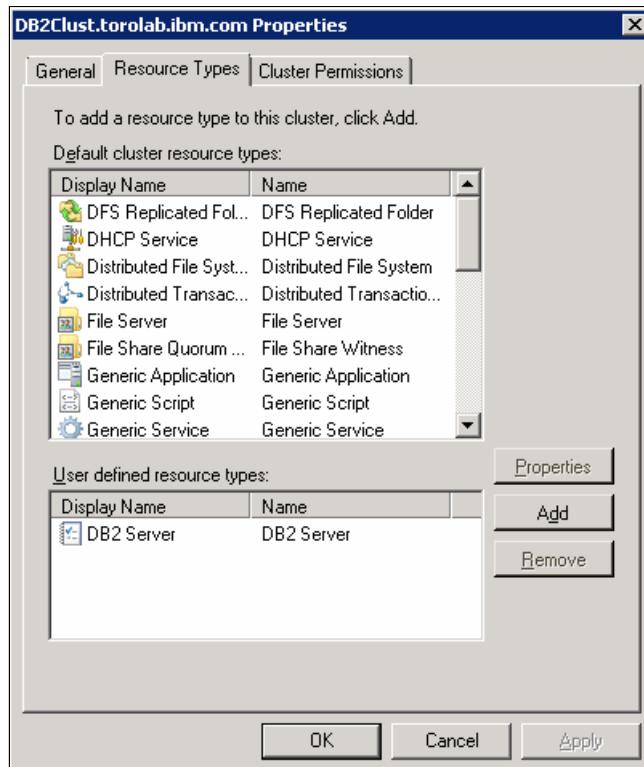


Figure 4-16 Resource types with DB2 Server resource type added

4.6.2 Creating cluster resources

To configure an application or service to be highly available, Failover Cluster creates a resource group for that application to hold all the needed resources. Failover and fallback act on that group. You can add every resource independently or use the Configure a Service or Application wizard to create them all for you. For simplicity, we use the Configure a Service or Application wizard for our example. From the right side bar, under Actions, click **Configure a Service or Application**.

The wizard takes you through a few simple steps to configure an application or service (DB2 in your case) to be highly available:

1. After you click **Configure a Service or Application**, a window opens (Figure 4-17) with an introduction about this wizard. Click **Next**.

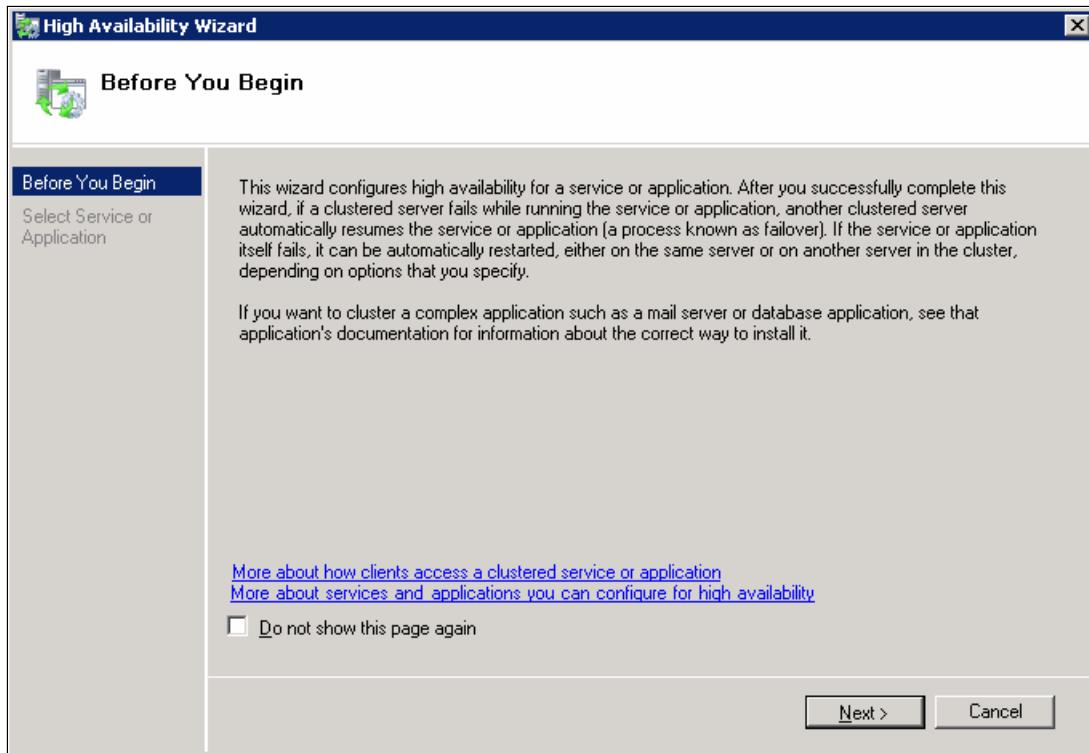


Figure 4-17 Configuring a service or application wizard

- The wizard prompts you for the Service or Application you want to make highly available. DB2 Server is not in the list, as it is a user-defined resource type. You can find Other Server in the list (Figure 4-18). Click **Other Server** and click **Next**.

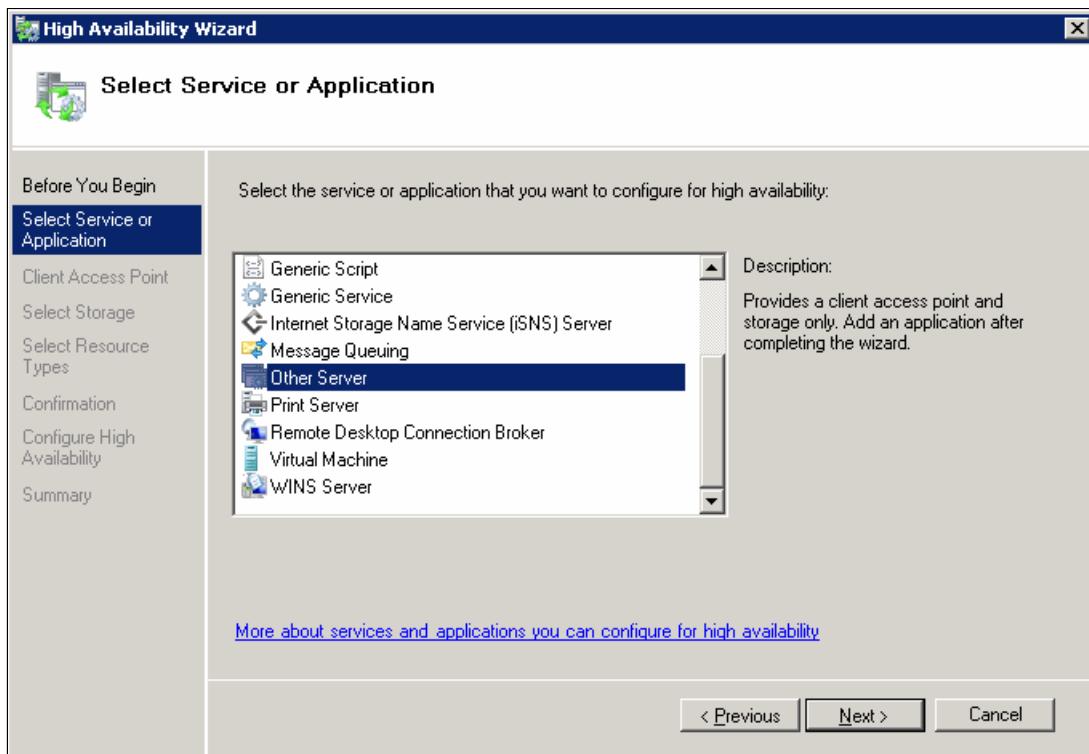


Figure 4-18 Select Service or Application

Other Server selection: After you select **Other Server** from the list, the number of wizard steps that are shown on the left jumped from two (Figure 4-17) to eight (Figure 4-18).

- The Client Access Point step is equivalent to creating a group and adding a highly available IP address to that group. Your client can later use this IP or access point name to connect to the application (DB2).
- The Select Storage step is equivalent to adding storage to your Service or Application group.
- The Select Resource Types step is where you tell Failover Cluster to create a resource of the DB2 Server type.

3. Now the Client Access Point window opens and prompts you for the access point name and IP address (Figure 4-19). Ensure that the name and address you select are not reserved by any other resource in the network. Otherwise, you get an error message and you must correct the error to proceed. Click **Next**.

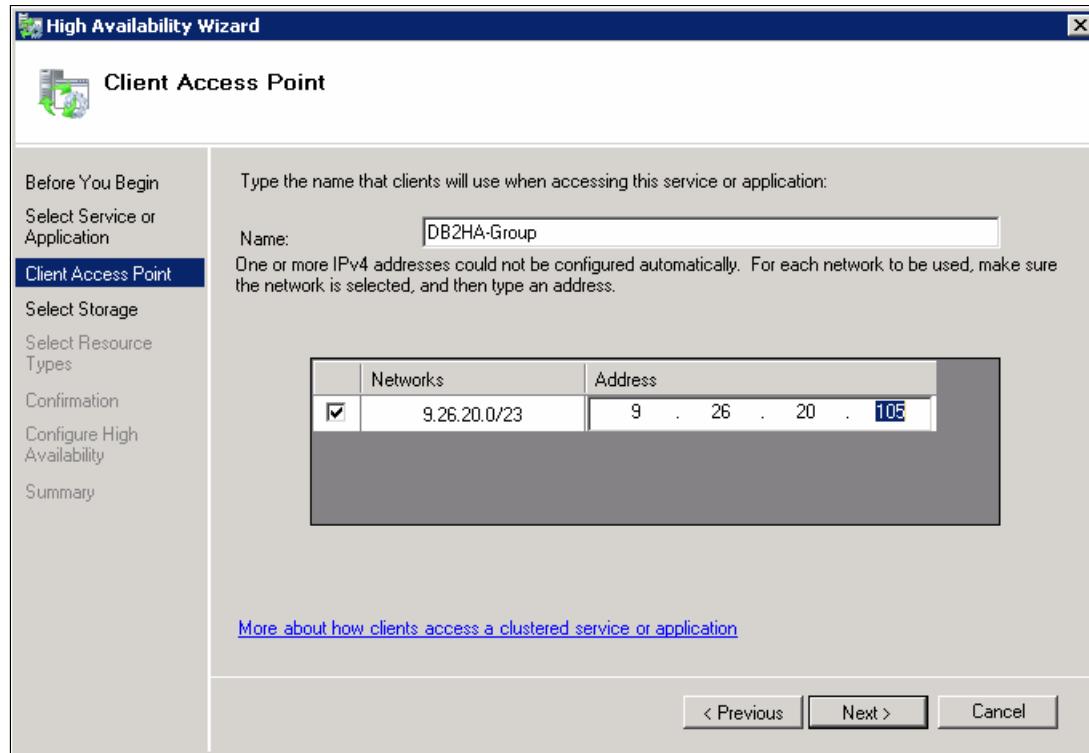


Figure 4-19 Client Access Point

4. Select the shared disk to store your data (Figure 4-20). By data, we mean the instance profile and your database files. Click **Next**.

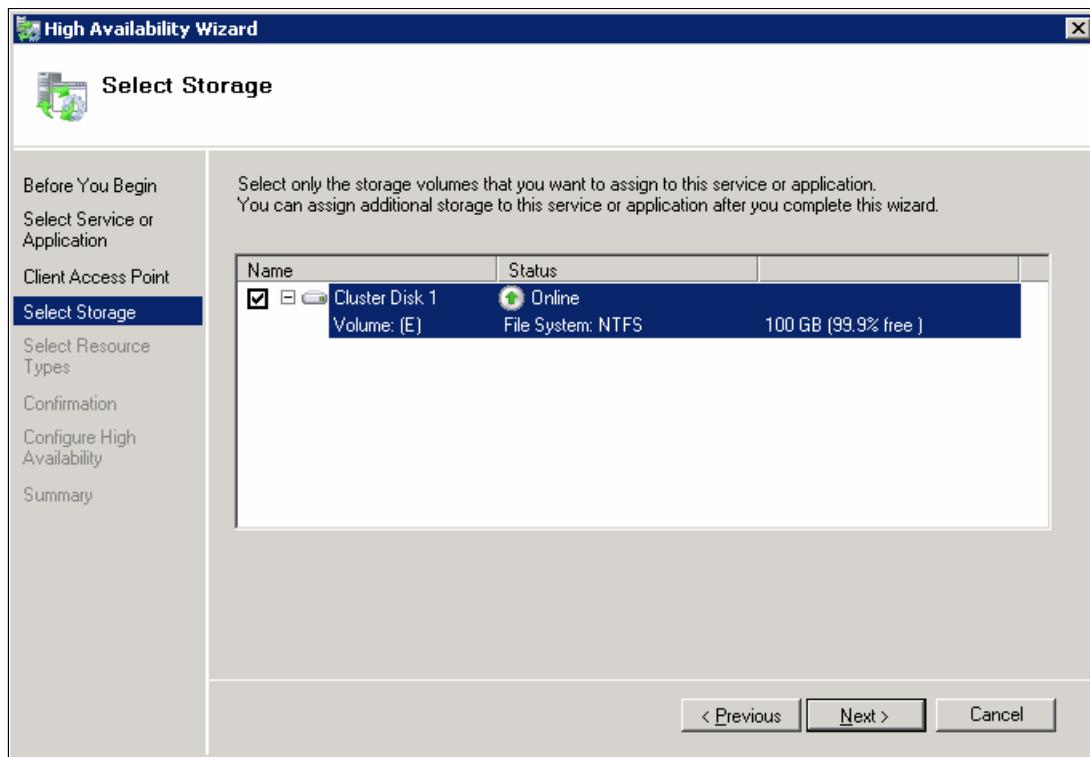


Figure 4-20 Select Storage

5. After this wizard completes successfully, Failover Cluster creates a resource of the resource type you select under this service or application group. Here we tell Failover Cluster to create a resource of type DB2 Server (Figure 4-21). If you have other user-defined resource types, you can find them in the list. Ensure that you select the **DB2 Server** resource type. Click **Next**.

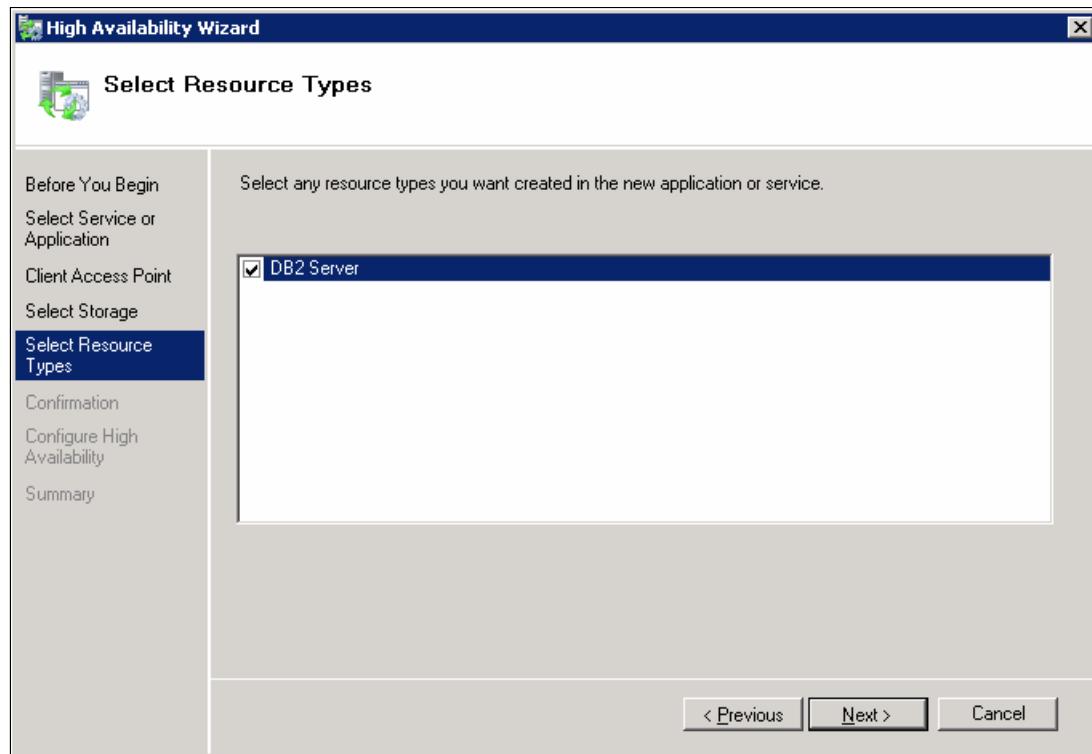


Figure 4-21 Select DB2 Server resource type

- Now you have provided all the necessary information. A confirmation window opens and lists all the information (Figure 4-22). Verify that this information is correct. If you must correct something, click **Previous** and make the corrections. After you are done, click **Next** to start configuring your application.

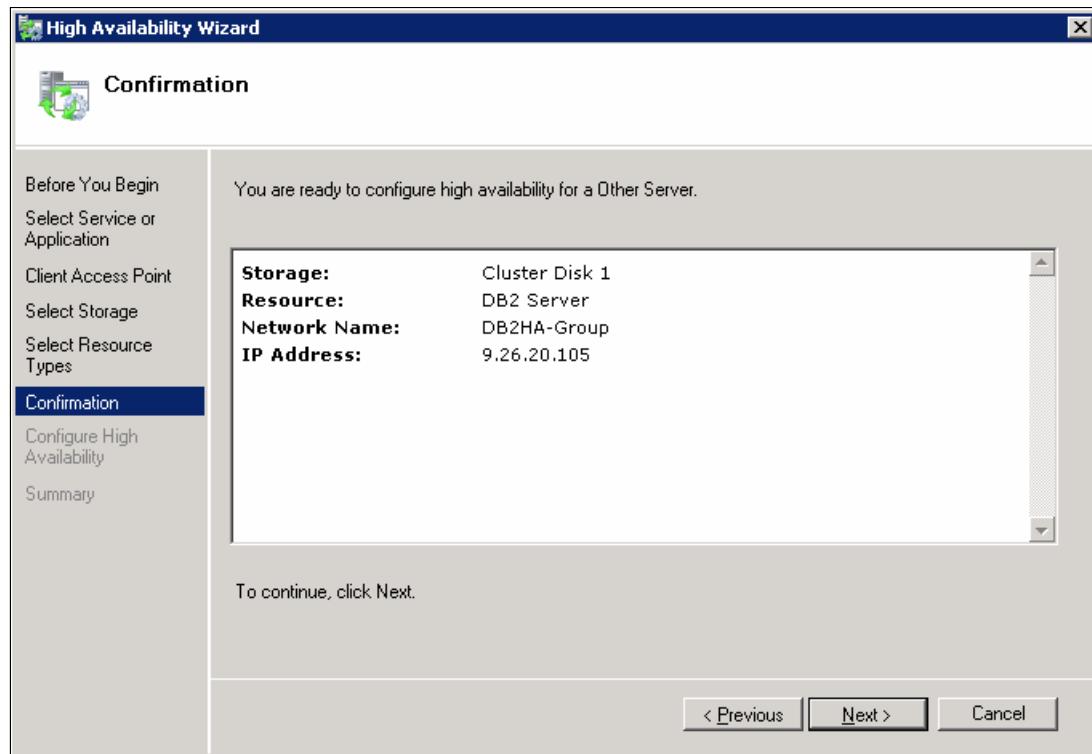


Figure 4-22 Confirmation window

7. The wizard configures the resources for you and shows you the progress (Figure 4-23).

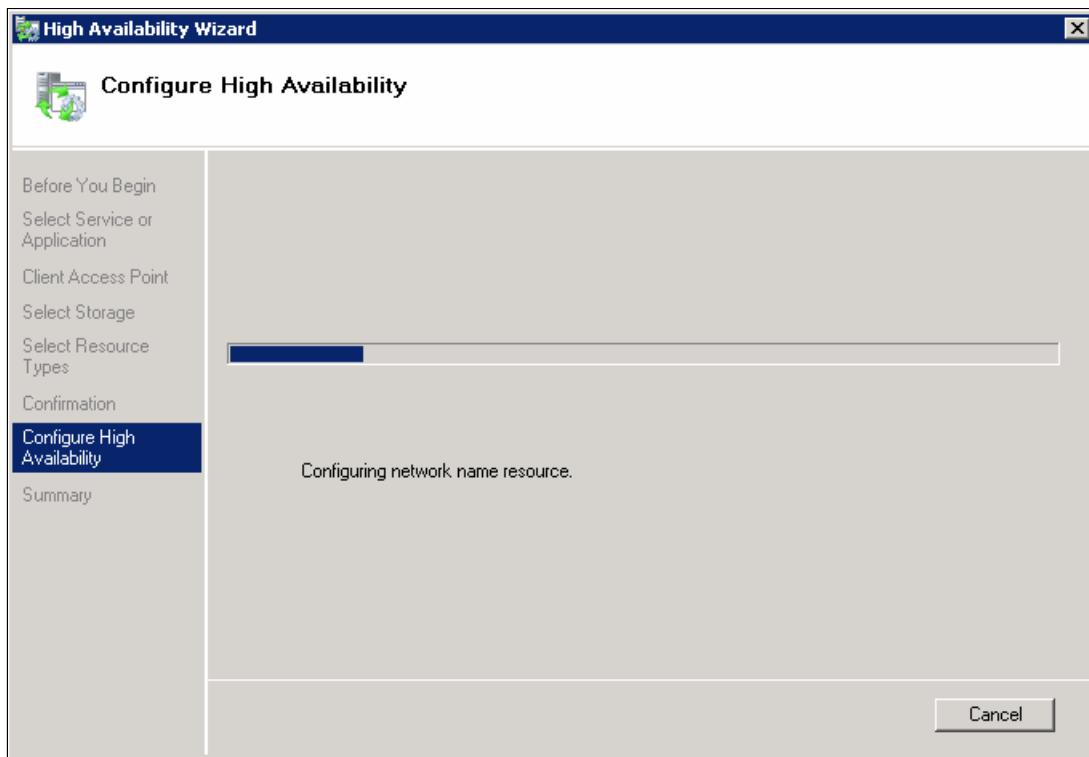


Figure 4-23 Configuring High Availability

8. After the previous step completes successfully, the wizard presents a summary report (Figure 4-24).

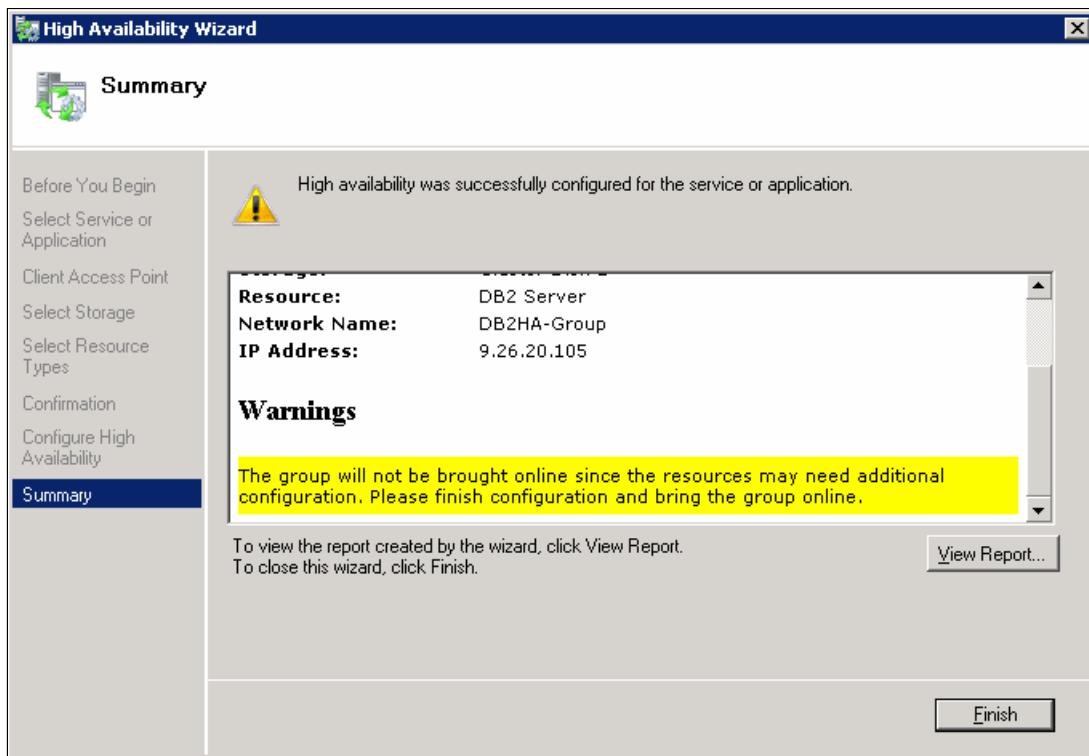


Figure 4-24 Summary report

Failover Cluster is not able to bring all resources online because more configuration is needed. Click **Finish** to see your created resource (Figure 4-25).

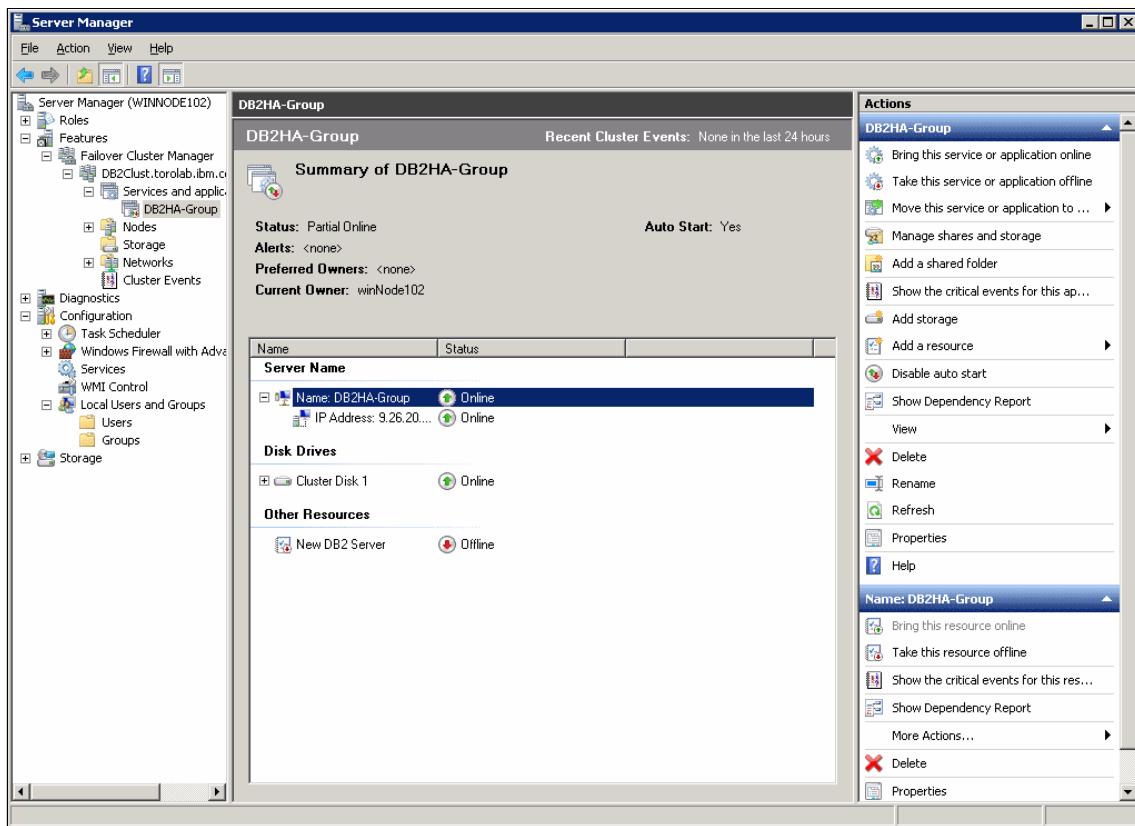


Figure 4-25 Newly configured service

The DB2 Server resource has the name “New DB2 Server”. You make the correct configurations to this resource later. Now you create a shared folder that is the instance profile directory to the server.

Multiple partition instances: When a multiple partition instance is running on a Windows Failover Cluster, the INSTPROF path must be set to a network path (for example, \\NetName\\DB2MSCS-DB2\\DB2PROFS). If you are using a single partition ESE instance, set the instance profile to a network path. This action is done automatically if you use the db2mscs command to cluster the DB2 database system.

9. To add a shared folder, complete the following steps:
 - a. Create a folder named DB2HA-0-Prof under the shared disk added to your group.
 - b. Right-click your cluster (DB2Clust) and click **Add a shared folder**. Failover Cluster examines your storage in the group and opens the Provision a Shared Folder wizard (Figure 4-26).

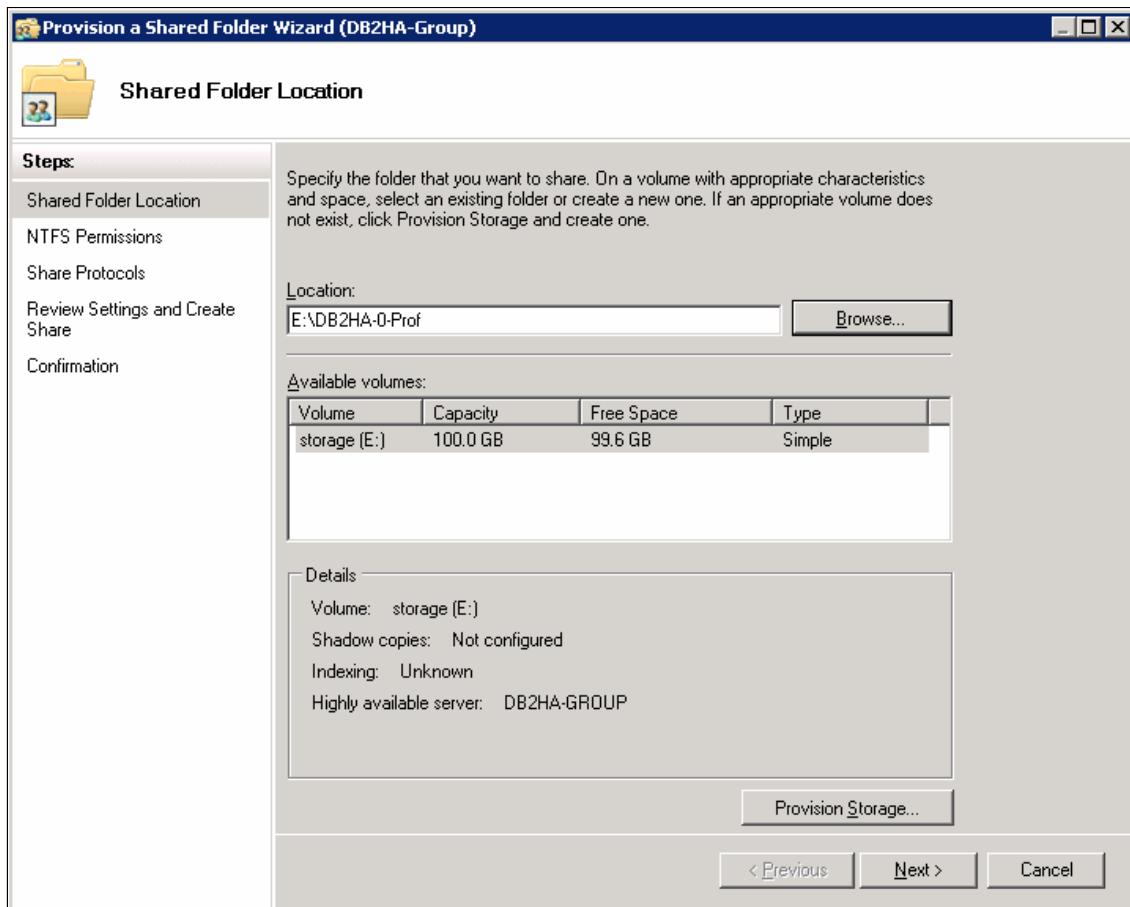


Figure 4-26 Provision A Shared Folder wizard

- c. Select the location of folder to be shared (E:\DB2HA-0-Prof) and click **Next**.
- d. Set the NTFS Permissions and click **Next**. In our example, we use:
 - TOROLAB\db2admn: Full control
 - TOROLAB\db2users: Change and Read

- Administrators: Full control
 - CREATOR OWNER: Full control
 - SYSTEM: Full control
 - LOCAL SERVICE: Full control
 - NETWORK SERVICES: Full control
- e. Under share name for SMB, enter DB2HA-0-Prof and click **Next** (Figure 4-27).

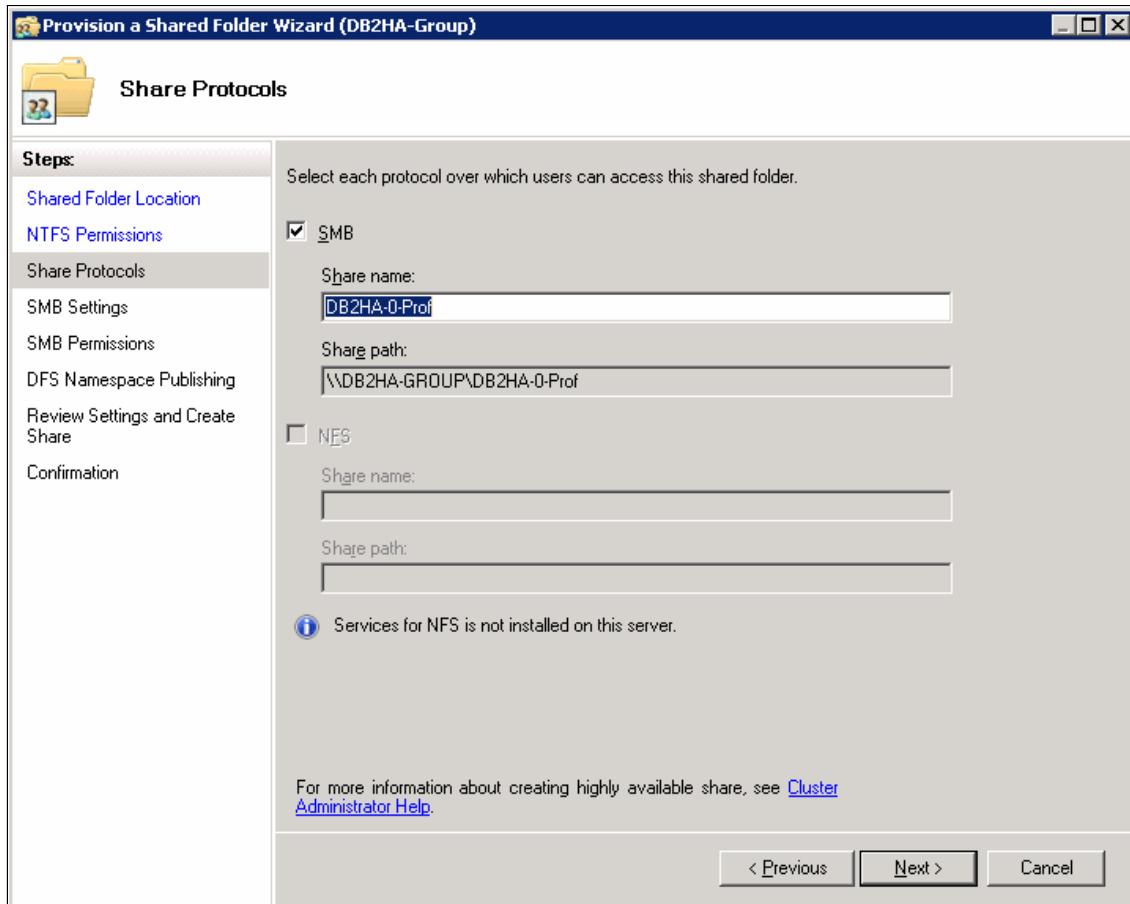


Figure 4-27 Select Share protocol and share name

- f. Click **Next** multiple times in the windows that open until you get a successful confirmation window (Figure 4-28). You are not required to make changes except in the SBM permissions window. You must set the correct permissions as we did in the NTFS permission. Without the correct permissions, you cannot migrate your instance to the cluster.

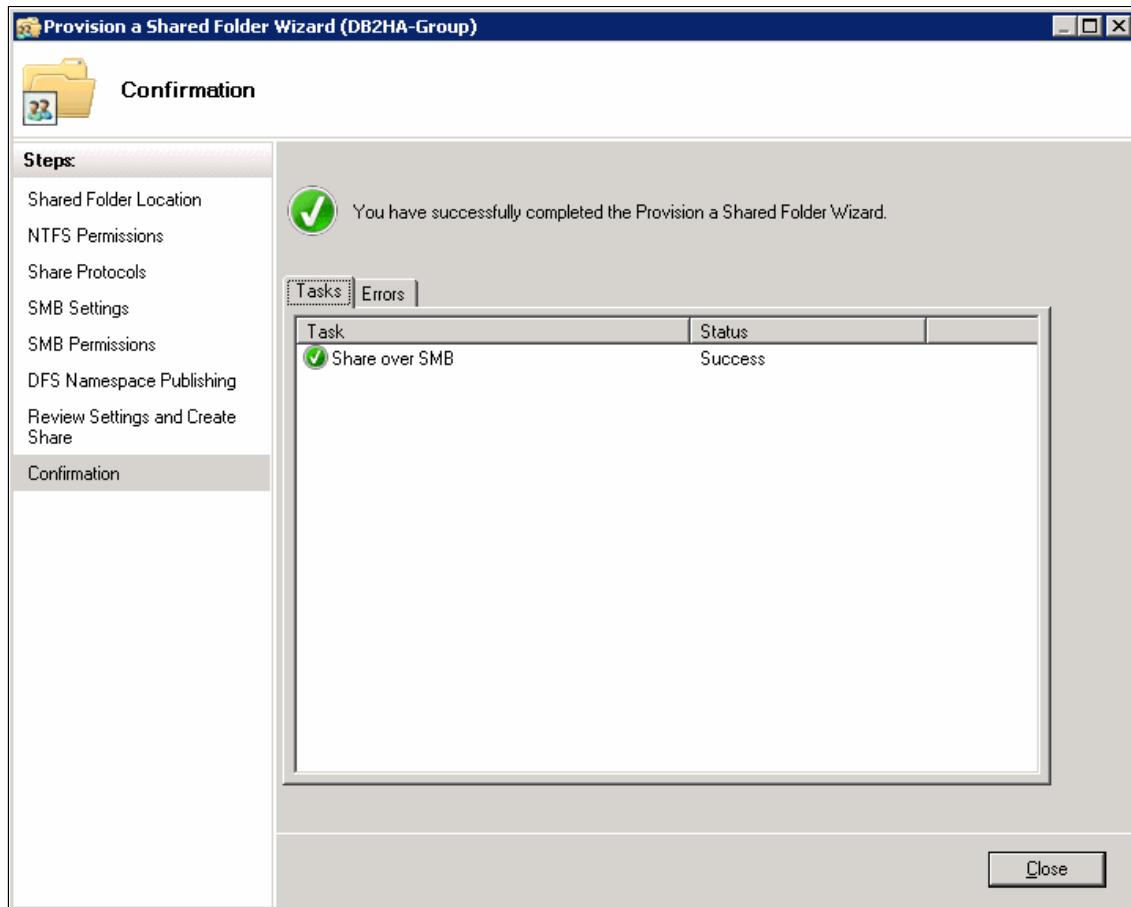


Figure 4-28 Successfully added shared folder

Now configure the DB2 Server resource (“New DB2 Server”). Open its properties by selecting it and clicking **Properties** from the bottom badge in the right side bar. Alternatively, you may right-click it and click **Properties**. From the Properties window, change the resource name to DB2HA-0 (Figure 4-29).

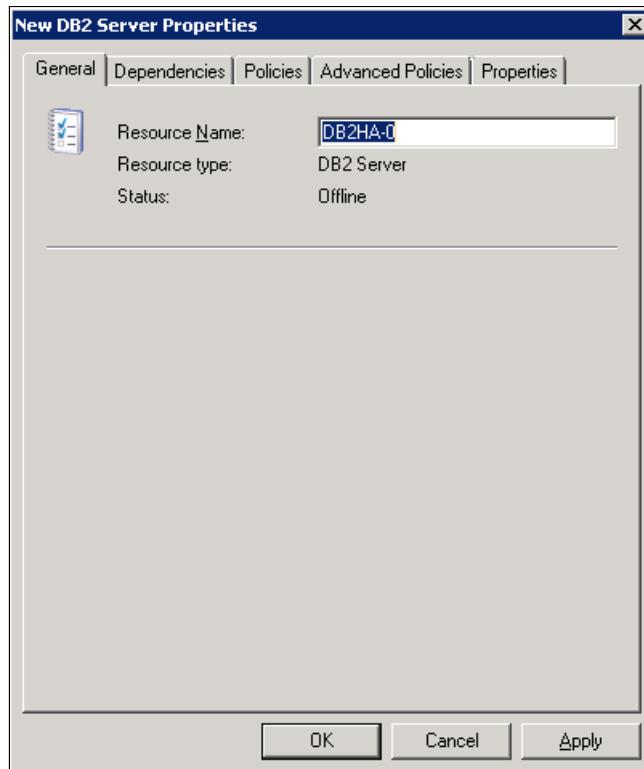


Figure 4-29 DB2 Server Resource Properties

By default, the Configure a Service or Application wizard set the dependencies of the DB2 Server resource. It adds all the resources except for the file server. Ensure all the dependencies, including the file server, are set correctly (Figure 4-30).

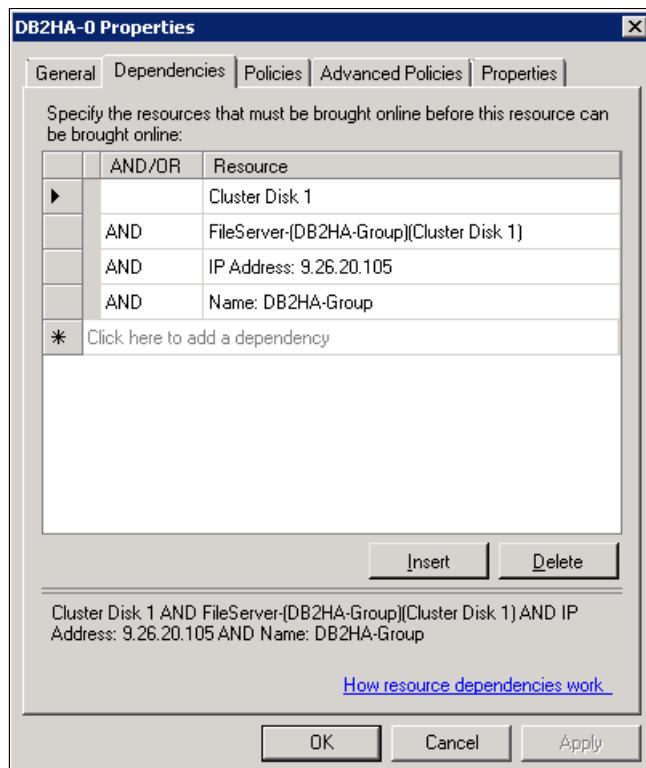


Figure 4-30 DB2 Server resource (DB2HA-0) dependencies

Section 4.5, “Creating a DB2 instance” on page 121 shows how to create a sample ESE instance. If you do not have a DB instance yet, you should create it now.

4.6.3 Migrating the DB2 instance to the cluster environment

The instance is in only one node in the cluster. If there is a failure, the other node cannot find information about this instance. The same situation applies to any other resource in the cluster. Windows Failover Cluster maintains information about the status, resources, resource types, and nodes of the cluster in the cluster database. This database exists in each node and the witness disk. It is synchronized constantly by the Failover Cluster software.

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 registry key in the cluster section of the Windows registry.

These two tasks are accomplished by running **db2iclus**, which must be run in the node where the DB2 instance was created.

In Figure 4-31, you can see a series of commands:

- ▶ **db2ilist** shows you that there is one instance in the system and it is not available in the cluster.³
- ▶ **db2iclus MIGRATE /i:DB2HA /c:DB2Clust /p:\DB2HA-Group\DB2HA-0-Prof /u:TOROLAB\anas.mosaad**: The **db2iclus** command moves the DB2HA instance to the DB2Clust cluster and places the DB2 instance profile directory in to the R:\DB2HAProf directory.
- ▶ **db2ilist** again shows you that the instance is now available in the DB2Clust cluster.

```
C:\IBM\SQLLIB\BIN>db2ilist
DB2HA

C:\IBM\SQLLIB\BIN>db2iclus MIGRATE /i:DB2HA /p:\DB2HA-Group\DB2HA-0-Prof /u:TOROLAB\anas.mosaad
Enter current password for TOROLAB\anas.mosaad:
DBI1912I The DB2 Cluster command was successful.

Explanation:
The user request was successfully processed.

User response:
No action required.

C:\IBM\SQLLIB\BIN>db2ilist
DB2HA          C : DB2Clust
C:\IBM\SQLLIB\BIN>_
```

Figure 4-31 Manually configuring a DB2 instance

³ We manually dropped the DB2 instance on to both nodes, which is why it is not showing in the command's output.

4.6.4 Adding a reference to the instance in the other nodes

The **db2iclus migrate** command places the DB2 instance profile directory in to the shared drive and adds the instance to the shared part of the Windows registry. However, the instance was created in one node only (winNode102 in our example). The instance creation process modifies several Windows registry keys and creates the service to be started in Windows. All these elements are missing from the other nodes in the cluster. If a failure occurs, Failover Cluster can correctly fail over to the other node (winNode101), but Windows does not find the Windows Service for the DB2 instance DB2HA. The failover process is ended.

Run **db2iclus** with the **ADD** clause to create references to the instance in the remaining nodes. In our example, the second node is winNode101. The command must be run from the node that has the instance to be referenced (winNode102) and is as follows:

```
db2iclus ADD /i:DB2HA /c:DB2Clust /p:\DB2HA-Group\DB2HA-0-Prof  
/u:T0R0LAB\anas.mosaad /m:winNode101
```

Figure 4-32 shows that the **db2ilist** command is run twice in winNode101. The second execution is after you run the **db2iclus ADD** command, which shows that the instance is now in this node.



```
C:\IBM\SQLLIB\BIN>db2ilist  
C:\IBM\SQLLIB\BIN>db2ilist  
DB2HA      C : DB2Clust  
C:\IBM\SQLLIB\BIN>_
```

Figure 4-32 DB2 instances list on winNode101

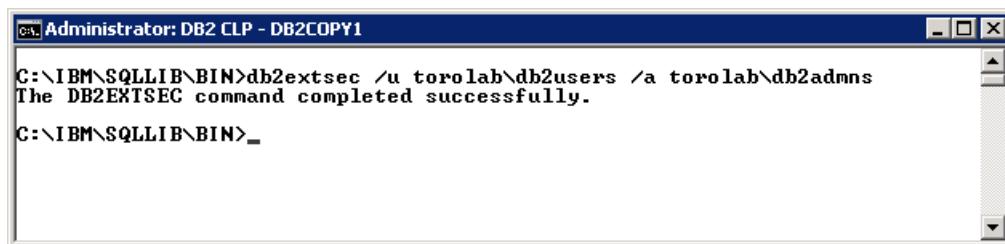
Ensure that the correct ports are added in the `\system32\drivers\etc\services` file.

4.6.5 Configuring security settings

By now, we have added to the cluster all the resources needed and the configured DB2 instance on the cluster. The files under the DB2 instance profile are protected by DB2 Access Control List. If the DB2_EXTSECURITY registry variable is set to YES on a Windows Failover Cluster, the DB2ADMNS and DB2USERS groups must be domain groups. You can review your DB2 global security configurations can under the HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE Windows registry key. Ensure that DB2_ADMINGROUP and DB2_USERSGROUP are domain groups, not local groups.

Using the db2extsec utility

db2extsec is used to set up the permissions for DB2 database objects (for example, files, directories, network shares, registry keys, and services) on updated DB2 database system installations. Use the utility to configure DB2 extended security. The command is used on the DB2 users and DB2 administrators groups (Figure 4-33).



```
C:\IBM\SQLLIB\BIN>db2extsec /u torolab\db2users /a torolab\db2admns
The DB2EXTSEC command completed successfully.

C:\IBM\SQLLIB\BIN>
```

Figure 4-33 Configuring DB2 extended security

Extended security: You must configure the extended security on both nodes of the cluster. Otherwise, the cluster may start on one machine and fails to start on the other one.

Setting the correct cluster permission

Failover Cluster uses your cluster name to create a computer object in the active directory of the domain. In our example, it is DB2Clust. This Cluster object (DB2Clust) must access DB2 files. Therefore, for the DB2 Server resource (DB2HA-0) to successfully start the instance on both nodes, this domain object (DB2Clust) must be added to the DB2ADMNS group on the domain.

Lastly, after you add the computer object to the DB2ADMNS group, you must reboot both nodes in the cluster. After reboot, your cluster is ready (Figure 4-34).

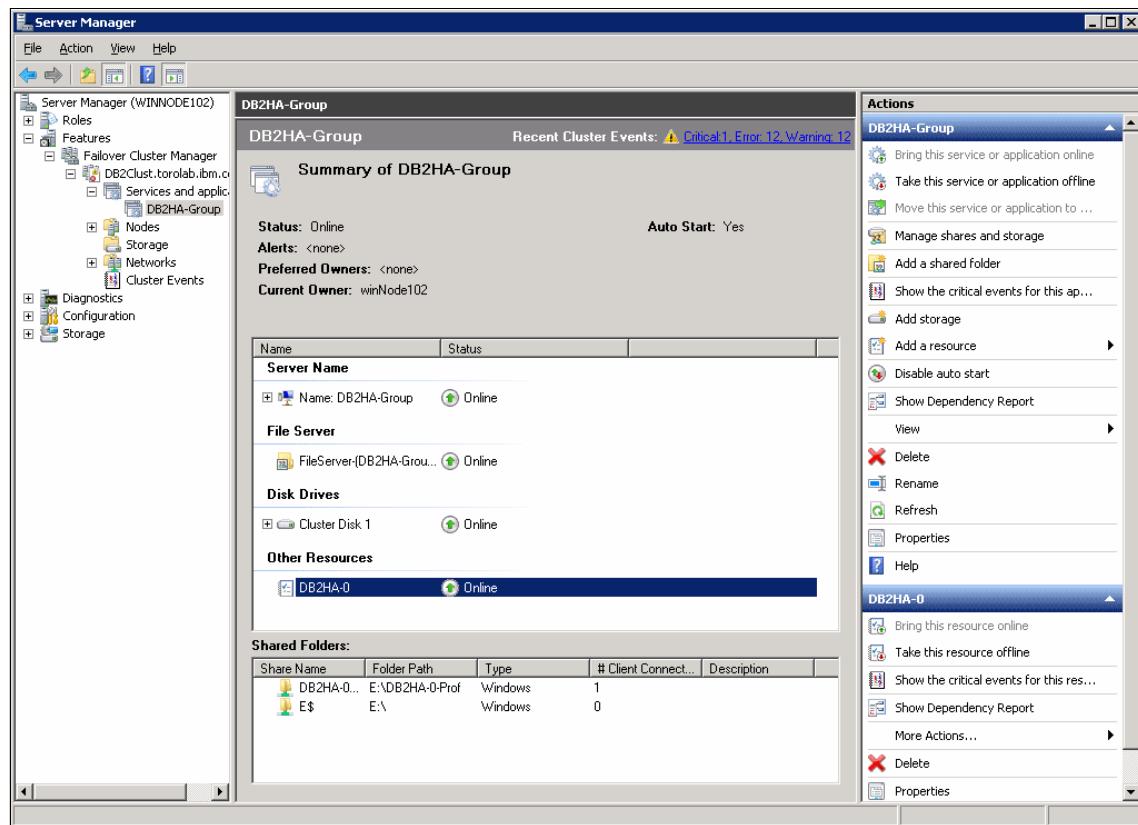


Figure 4-34 Cluster group DB2HA Group up and running

Domain changes: The domain changes are not replicated to the local machines until you restart it, which is why you must reboot both nodes you add the Cluster object to the DB2ADMNS group.

4.7 Using db2mscs to configure a DB2 instance

The **db2mscs** utility automates most of the steps of creating a highly available DB2 instance in the Failover Cluster environment. The prerequisite is that the shared disk where the instance profile is moved to must be made highly available before you run 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, such as IP addresses and network names. **db2mscs** accepts three parameters:

```
db2mscs -f:<config_file> -u:<instance name> -d:<debug_file>
```

Where:

- ▶ **-f:** This option specifies the configuration file. The default value of this parameter is `db2mscs.cfg`.
- ▶ **-u:** This option reverts all changes in the specified instance and returns it to its unclustered form.
- ▶ **-d:** This option specifies the debug file name. The utility writes debugging information in to 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 `%DB2Dir%\sqllib\cfg` directory:

- ▶ `db2mscs.das` (Clusters a DAS instance.)
- ▶ `db2mscs.ee` (Clusters an EE instance.)
- ▶ `db2mscs.eee` (Clusters a DPF instance.)

Because we created an ESE instance in 4.5, “Creating a DB2 instance” on page 121, we demonstrate using **db2mscs** to configure a Workgroup Server Edition (WSE) instance.

- ▶ Figure 4-35 on page 145 shows the commands to create a DB2 WSE instance named DB2WSE. If you are concerned about saving your password in a local text file, you can drop or comment out the `DB2_LOGON_PASSWORD` property and **db2mscs** prompts you for the user password at run time. For more information, see the DB2 Information Center at <http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.cmd.doc%2Fdoc%2Fr0002078.html> and or *DB2 10.1 Command Reference*, SC27-3868-00.

```
C:\IBM\SQLLIB\BIN>db2ilist
DB2HA      C : DB2Clust

C:\IBM\SQLLIB\BIN>db2icrt DB2WSE -s WSE -u torolab\anas.mosaad
Enter current password for torolab\anas.mosaad:
DB20000I  The DB2ICRT command completed successfully.

C:\IBM\SQLLIB\BIN>db2ilist
DB2WSE      C : DB2Clust

C:\IBM\SQLLIB\BIN>_
```

Figure 4-35 Creating Workgroup instance DB2WSE

The sample configuration file that we use is shown in Example 4-1. The **DB2_LOGON_PASSWORD** property is commented out.

Example 4-1 Sample Failover Cluster configuration file

```
DB2_LOGON_USERNAME=torolab\anas.mosaad
#DB2_LOGON_PASSWORD=xxxxxxxxxx
CLUSTER_NAME=DB2Clust
GROUP_NAME=DB2WSE Group
DB2_INSTANCE=DB2WSE

IP_NAME=DB2WSE IP Address
IP_ADDRESS=9.26.20.107
IP_SUBNET=255.255.255.0
IP_NETWORK=Cluster Network 2

NETNAME_NAME=DB2WSE Net
NETNAME_VALUE=DB2WSE-Public
NETNAME_DEPENDENCY=DB2WSE IP Address

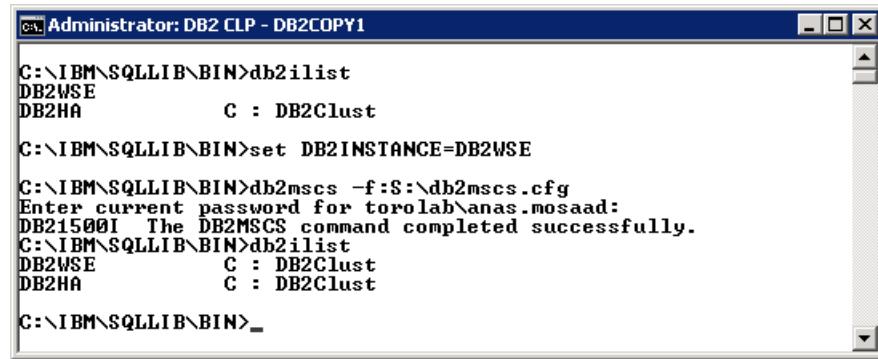
DISK_NAME=Disk S
INSTPROF_PATH=S:\DB2WSEProf
```

Here are some points to notice in this example:

- ▶ The value of **IP_NAME** is arbitrary. Use the instance name for the **IP_NAME**.
- ▶ The value of **NETNAME_DEPENDENCY** must 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 exists in the cluster.

Before you run **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 task, the operation fails because **db2mscs** tries to start the instance that is specified by **DB2INSTANCE**.

Changing the instance name solves the problem (Figure 4-36).



```
C:\Administrator: DB2 CLP - DB2COPY1
C:\IBM\SQLLIB\BIN>db2ilist
DB2WSE
DB2HA      C : DB2Clust

C:\IBM\SQLLIB\BIN>set DB2INSTANCE=DB2WSE

C:\IBM\SQLLIB\BIN>db2mscs -f:S:\db2mscs.cfg
Enter current password for torolab\anas.mosaad:
DB21500I The DB2MSCS command completed successfully.

C:\IBM\SQLLIB\BIN>db2ilist
DB2WSE      C : DB2Clust
DB2HA      C : DB2Clust

C:\IBM\SQLLIB\BIN>_
```

Figure 4-36 Configuring the DB2 instance using db2mscs

Your Workgroup DB2 instance DB2WSE is now clustered on your DB2Clust cluster. If you open the Failover Cluster management console, you find the resources that are created and configured: a group, the highly available IP address, storage added, and the DB2 Server resource. It also starts the group and starts all its resources (Figure 4-37).

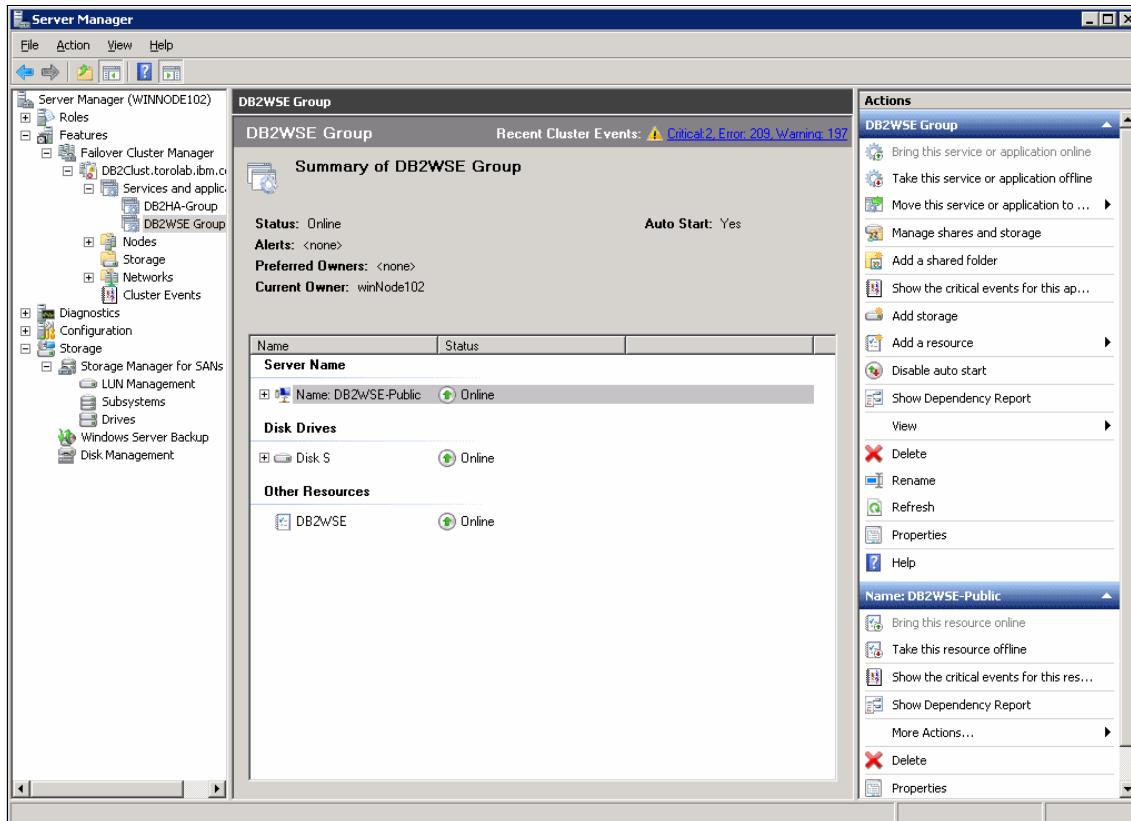


Figure 4-37 DB2WSE instance clustered using db2mscs

Important: Ensure that you complete the steps in 4.6.5, “Configuring security settings” on page 142. Those steps must be applied whether you configured the instance manually or by using the `db2mscs` utility.

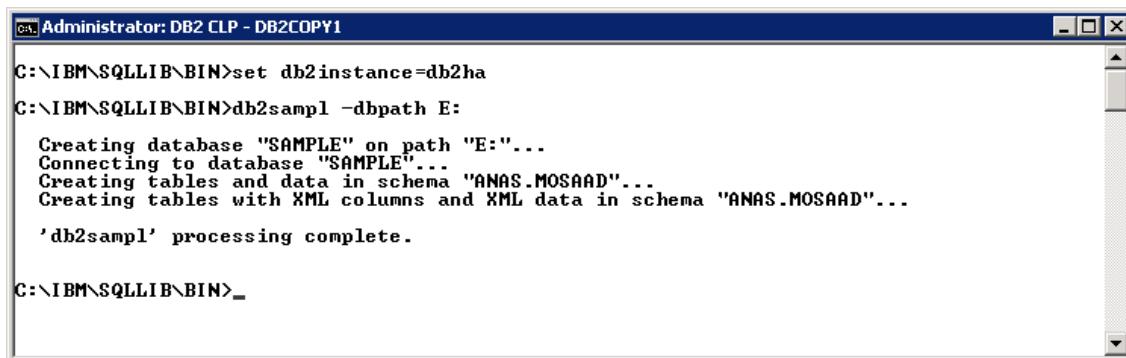
4.8 Testing a cluster

Generally, you do clustering to eliminate single points of failure. Testing such a setup requires a good testing plan to test all possible points of failure. We are not going to describe a test plan here, but do tell you how to test the DB2 part. Here are the general steps:

1. Create a database and ensure that it is stored on the shared storage. If you put your database on a new shared disk, ensure to add this disk to your cluster group for your database instance.
2. From your application, connect to the database using the highly available IP.
3. Run some SQL statements. They should run successfully.
4. Fail over your DB2 group (DB2HA-Group) to the second node.
5. Using the same connection, try to issue the same SQL statements again. If the queries failed, reconnect again using the same IP address. Issue the statements and the failover should succeed.
6. Fail back the group to the primary node.
7. From your application, using the current connection, run the SQL statements. If the statements fail, reconnect and try again. They should succeed.
8. Repeat the previous steps many times while you simulate various hardware and software failures.

4.8.1 Creating a SAMPLE database

First, create a sample database by using the **db2samp1** utility. Before you run the **db2samp1** utility, ensure that you are running it from the current owner of DB2 cluster group, the group is online, you specified the dbpath on the shared storage, and you set the **DB2INSTANCE** environment variable with the correct instance name (Figure 4-38).



```
C:\IBM\SQLLIB\BIN>set db2instance=db2ha
C:\IBM\SQLLIB\BIN>db2samp1 -dbpath E:
Creating database "SAMPLE" on path "E:"...
Connecting to database "SAMPLE"...
Creating tables and data in schema "ANAS.MOSAAD"...
Creating tables with XML columns and XML data in schema "ANAS.MOSAAD"...
'db2samp1' processing complete.

C:\IBM\SQLLIB\BIN>_
```

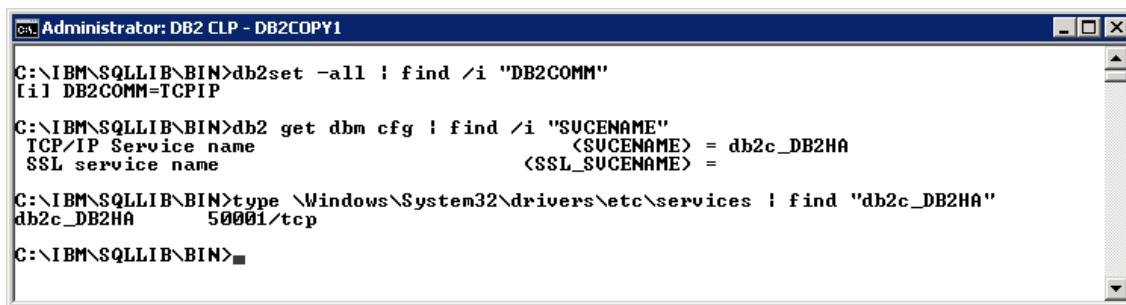
Figure 4-38 Creating the SAMPLE database on the shared storage

4.8.2 Verifying the DB2 instance communication settings

Before you try to connect to the database from a remote client, you must ensure that TCP/IP communication is enabled on the instance and a valid communication port is configured on both nodes. Here are the checklist items that you must verify:

- ▶ Verify that the instance is configured to start listening on the appropriate protocol by running **db2set -a11**. Look for the **DB2COMM** parameter.
- ▶ Verify that the appropriate protocol-specific parameters are set in the database manager configuration settings so that DB2 knows what values to listen on. For example, run **db2 get dbm cfg | find /i "SVCENAME"**.
- ▶ If the previous step resulted in a service name instead of a port number for the **SVCENAME** field in your environment, then confirm that the value listed there is mapped to a unique port number in the operating system's **%windir%\system32\drivers\etc\services** file. The configuration must be the same on both nodes.

Figure 4-39 shows a series of commands to verify the communication settings.



```
C:\IBM\SQLLIB\BIN>db2set -all | find /i "DB2COMM"
[i] DB2COMM=TCPPIP

C:\IBM\SQLLIB\BIN>db2 get dbm cfg | find /i "SUCENAME"
TCP/IP Service name           <SUCENAME> = db2c_DB2HA
SSL service name              <SSL_SUCENAME> =

C:\IBM\SQLLIB\BIN>type \Windows\System32\drivers\etc\services | find "db2c_DB2HA"
db2c_DB2HA      50001/tcp
C:\IBM\SQLLIB\BIN>
```

Figure 4-39 Verify communication settings

4.8.3 Connecting to the database using Data Studio

Connect to this SAMPLE database using the highly available IP address from Data Studio. Configure Data Studio 3.1.1 on Linux to connect to the sample database we created under the DB2HA instance. In Figure 4-19 on page 128, we assigned IP address 9.26.20.105 as the access point for this instance.

Open Data Studio and from the Administration Explorer view⁴, click **New → New Connection to a database**. A window opens (Figure 4-40). Enter the database connection parameters.

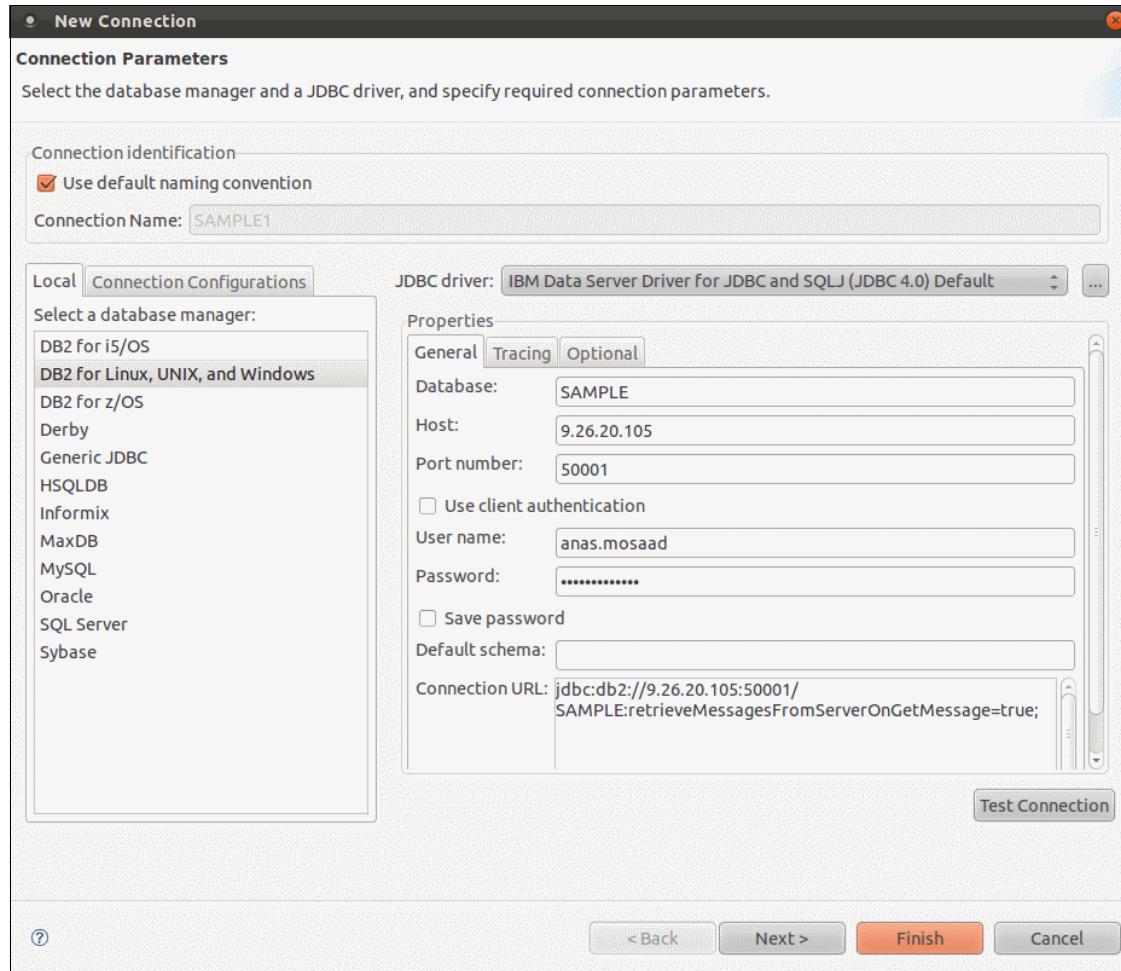


Figure 4-40 Creating a connection to the cluster database

⁴ If the Administration Explorer view is not already open, open the Show view window by clicking **Window → Show view → Other**. In the Show view window, select the view by clicking **Data Management → Administration Explorer** and clicking **OK**.

Ensure that your database connection parameters are correct and click **Test Connection**. You should get a successful connection message (Figure 4-41). Click **Finish** to add this database to your database list.

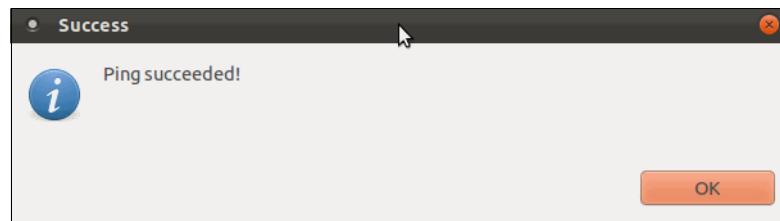


Figure 4-41 Connection successful

4.8.4 Testing failover

Data Studio added the database to your databases list and connected to it. You can run some SQL statements against the database to make sure that it is working correctly. Run an SQL query “`select deptno, deptname from "ANAS.MOSAAD".DEPARTMENT;`” to confirm that you can run queries successfully. You should get a successful result (Figure 4-42).

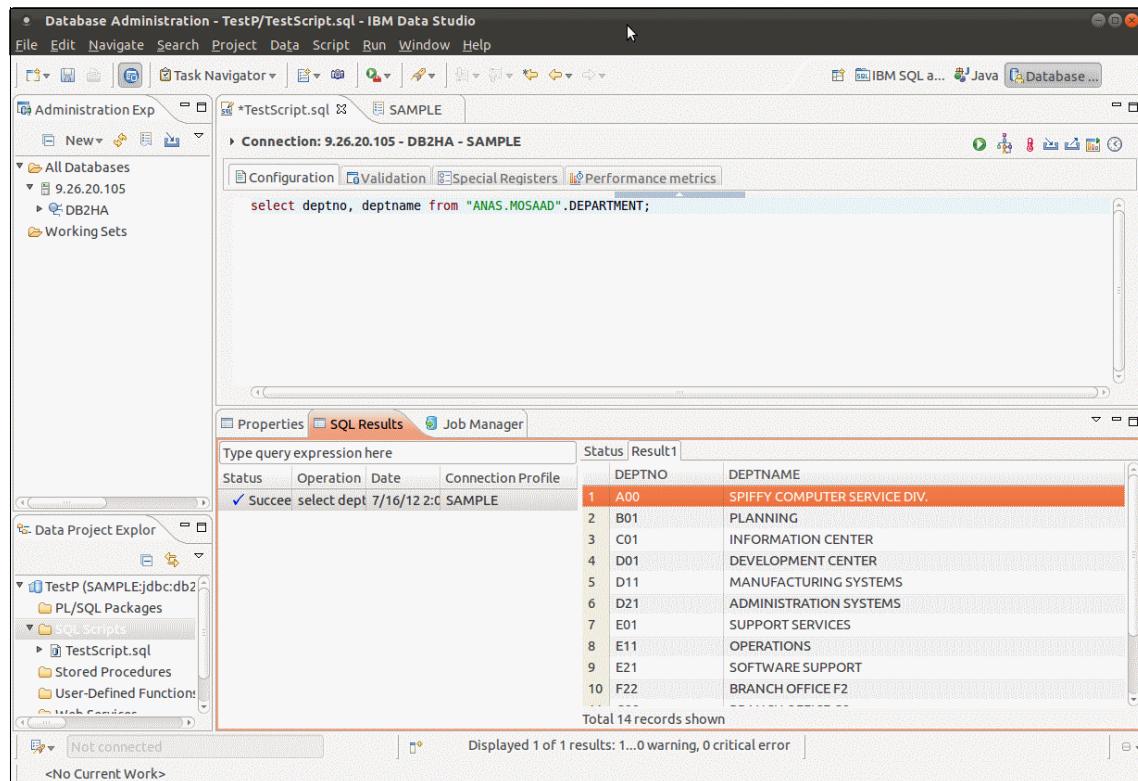


Figure 4-42 Successful query execution

Now move your cluster groups from the primary node to the secondary node. Data Studio automatically detects that your instance stopped. It displays a notification that automatic reconnection failed (Figure 4-43).

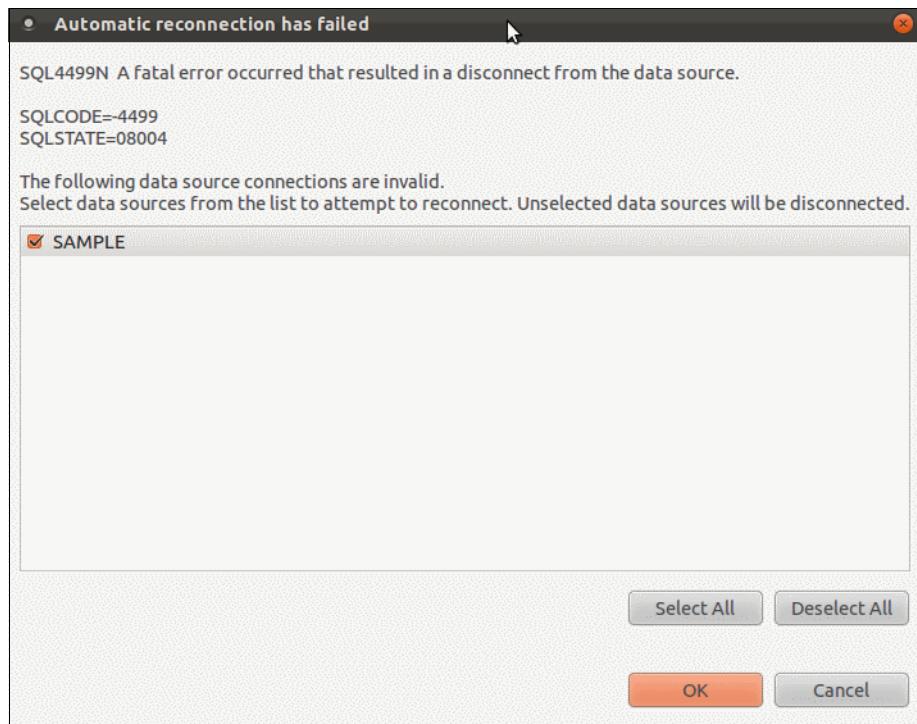


Figure 4-43 Data Studio automatic reconnection failure notification

This notification message automatically disappears after Data Studio is able to reconnect again to your database. Now try to run the same SQL statement again. It should succeed and you get the result shown in Figure 4-42 on page 153.

Repeat the same scenario after you move the cluster group to the primary node again. Run the same statement again; it should succeed.

If you have a long running transaction in your database, moving the group from one node to the other might fail. To ensure that such scenario is avoided, set the DB2 registry variable **DB2_FALLBACK** to on. To set **DB2_FALLBACK ON**, run **db2set DB2_FALLBACK=ON**.

Now you have a clustered DB2 instance up and running. You tested that it is working correctly on both nodes. You can use this environment to upgrade your DB2 instance.

4.9 Upgrading your instance

A clustered environment is ideal for performing a rolling upgrade to your servers. You are able to upgrade your database instance fix pack with minimal downtime. To do rolling upgrades of your servers, complete the following steps:

1. Have winNode101 as the active node and winNode102 as idle.
2. Stop the cluster server service on the idle node (winNode102).
3. Apply your fixes on winNode102. In our case, we apply a DB2 fix pack.
4. Take all DB2 Server resources offline.
5. Start the cluster service on the winNode102 node.
6. Move your clustered instance group to winNode102.
7. Bring the group online. winNode101 is now idle.
8. Repeat steps 2 - 5, but on winNode101 instead.
9. Move the group back to winNode101.

DPF environment: In a DPF environment, ensure that all active partitions are running the same db2level.

If you plan to upgrade your database version, see the Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.qb.upgrade.doc%2Fdoc%2Ft0022647.html>



DB2 HADR introduction

Organizations 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 was 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)*.

This chapter covers the following topics:

- ▶ HADR overview
- ▶ HADR architecture
- ▶ Terminology

5.1 HADR overview

The HADR feature provides a high availability solution for both partial and complete site failures. HADR protects against data loss by replicating data changes from a source database, called the primary database, to one or more target databases, called the standby databases.

A partial site failure can be caused by a hardware, network, or software (DB2 database system or operating system) failure. Without HADR, a partial site failure requires restarting the database management system (DBMS) server that contains the database. The length of time that it takes to restart the database and the server where it is located is unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. With HADR, a standby database can take over in seconds. Further, you can redirect the clients that used the original primary database to the new primary database by using automatic client reroute or retry logic in the application. This redirection is also possible using software solutions that are described in Chapter 10, “Automatic client reroute” on page 377, which can manage IP address redirection.

A complete site failure can occur when a disaster, such as a fire, causes the entire site to be destroyed. However, because HADR uses TCP/IP for communication between the primary and standby databases, these databases can be in different locations. For example, the primary database might be at your head office in one city, and a standby database might be at your sales office in another city. If a disaster occurs at the primary site, data availability is maintained by having the remote standby database take over as the primary database with full DB2 functionality. After a takeover operation occurs, you can bring the original primary database back up and return it to its primary database status; this concept is known as *failback*. You can initiate a failback if you can make the old primary database consistent with the new primary database. After you reintegrate the old primary database into the HADR setup as a standby database, you can switch the roles of the databases to enable the original primary database to once again be the primary database.

DB2 HADR includes all the following functions:

- ▶ Fast failover capability
- ▶ Negligible impact on performance
- ▶ Easy to set up and monitor
- ▶ Rolling upgrades with no downtime for running applications
- ▶ Transparent failover and failback for applications
- ▶ Easy integration with high availability clustering software
- ▶ Improved disaster recovery compared to conventional methods

Figure 5-1 illustrates the concept of HADR.

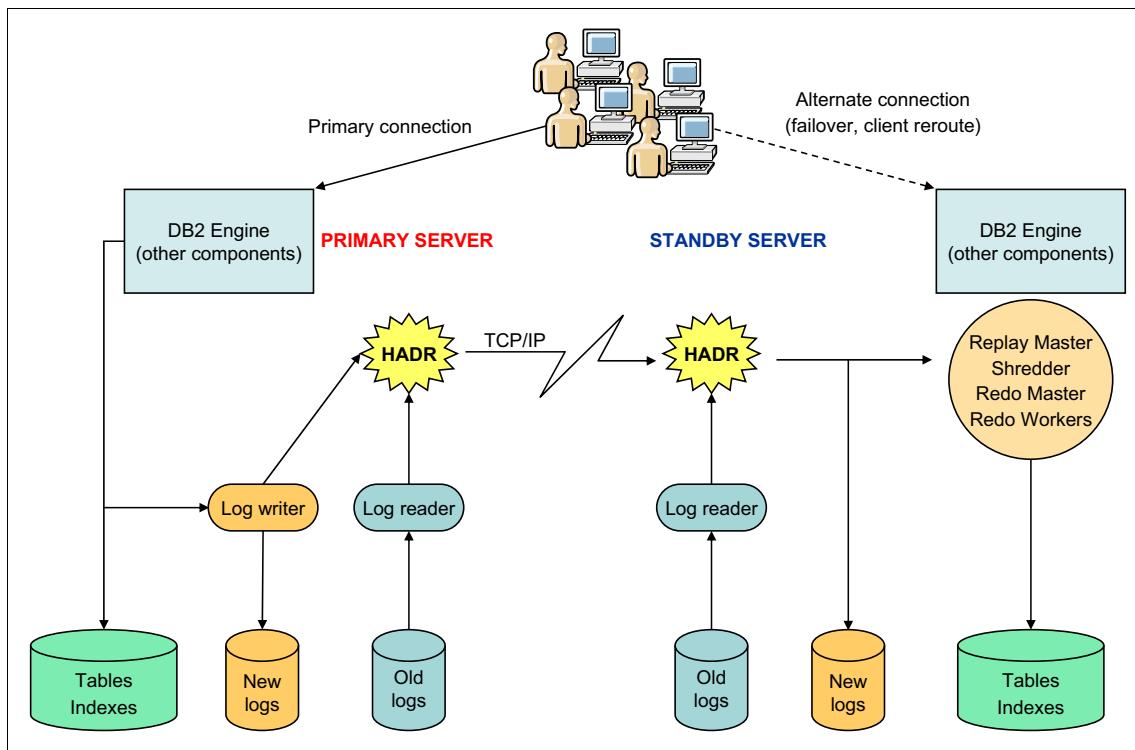


Figure 5-1 HADR overview

All changes that take place at the primary database server are written into log records. These log records are then shipped to the secondary database server where the recorded changes are replayed to the local copy of the database. This procedure ensures that the primary and the secondary database servers are in a synchronized state. From this point on, 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 fast. In DB2 10.1, it is also possible to use the reads on standby capability to run read-only operations on the standby database in your HADR solution. Read operations that are running on a standby do not affect the standby's main role of replaying logs that are shipped from the primary database.

Using two dedicated TCP/IP communication ports and a heartbeat, the primary and the standby databases track where they are processing currently, the current state of replication, and whether the standby database is up to date with the primary database (known as a HADR Peer state). When a log record 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, the HADR communication between the primary and the standby takes place over TCP/IP, which enables the database to be replicated to a remote geographical site. This setup allows the database to be recovered either locally or at the remote site if there is a disaster at the primary database server site. The HADR solution thus provides both High Availability and Disaster Recoverability. HADR provides an incomparable improvement on conventional methods for DB2 disaster recovery, which otherwise could mean losses in hours of committed transaction data.

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

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

5.1.1 HADR topology

With HADR, you base the level of protection from potential loss of data on your configuration and topology choices. Some of the key choices that you must make are as follows:

- ▶ What level of synchronization do you use?

Standby databases are synchronized with the primary database through log data that is generated on the primary and shipped to the standbys. The standbys constantly roll forward through the logs. You can choose from four different synchronization modes. In order of most to least protection, these modes are synchronous (SYNC), near-synchronous (NEARSYNC), asynchronous (ASYNC), and super-asynchronous (SUPERASYNC). For more information, see 5.1.2, “HADR synchronization modes” on page 164.

- ▶ Do you use a peer window?

The peer window feature specifies that the primary and standby databases are to behave as though they are still in the Peer state for a configured amount of time if the primary loses the HADR connection in the Peer state. If the primary database fails in the Peer state or this “disconnected peer” state, the failover to standby has zero data loss. This feature provides the greatest protection.

► How many standbys do you deploy?

With HADR, you can use either single standby mode or multiple standby mode. With multiple standbys, you can achieve both your high availability and disaster recovery objectives with a single technology.

There are many ways that you can use your HADR standby databases beyond their HA or DR purposes:

- Reads on standby feature

You can use the reads on standby feature to direct read-only workload to one or more standby databases without affecting the HA or DR responsibility of the standby database. This feature can reduce the workload on the primary database without affecting the main responsibility of the standby database.

Unless you have reads on standby enabled, applications can access the current primary database only. If you have reads on standby enabled, read-only applications can be redirected to the standby database.

Applications connecting to the standby database do not affect the availability of the standby database if there is a failover.

- Delayed replay feature

You can use the delayed replay feature to specify that a standby database should remain at an earlier point in time than the primary database, in terms of log replay. If data is lost or corrupted on the primary database, you can recover this data on the time delayed standby database.

- Rolling updates and upgrades feature

Using an HADR setup, you can make various types of upgrades and DB2 fix pack updates to your databases without an outage. If you have multiple standby modes that are enabled, you can perform an upgrade while you keep the protection provided by HADR.

HADR might be your best option if most or all data in your database requires protection or if you perform DDL operations that must be automatically replicated on a standby database. However, HADR is only one of several replication solutions that are offered in the DB2 Product Family. The InfoSphere Federation Server software and the DB2 database system include SQL replication and Q replication solutions that you can also use, in some configurations, to provide high availability. These solutions 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, and updates to any copy of a table. You can also use these solutions in partitioned database environments.

The standard topology is one primary and one standby database. This topology makes up the most commonly seen layout. This setup consists of a pair of servers, one with a single DB2 instance and a single database that acts as the primary database, and another with a single DB2 instance and a single database that acts as the standby database.

However, this situation is by no means the only 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. Therefore, a standby server can have multiple databases from multiple primaries on it. Another configuration option is to implement two servers in an active/active cluster mode (Figure 5-2). Here the HADR primary databases are spread across both servers in a load-sharing configuration. The advantage with this option is that it makes more efficient use of available processor cycles, and other valuable server resources.

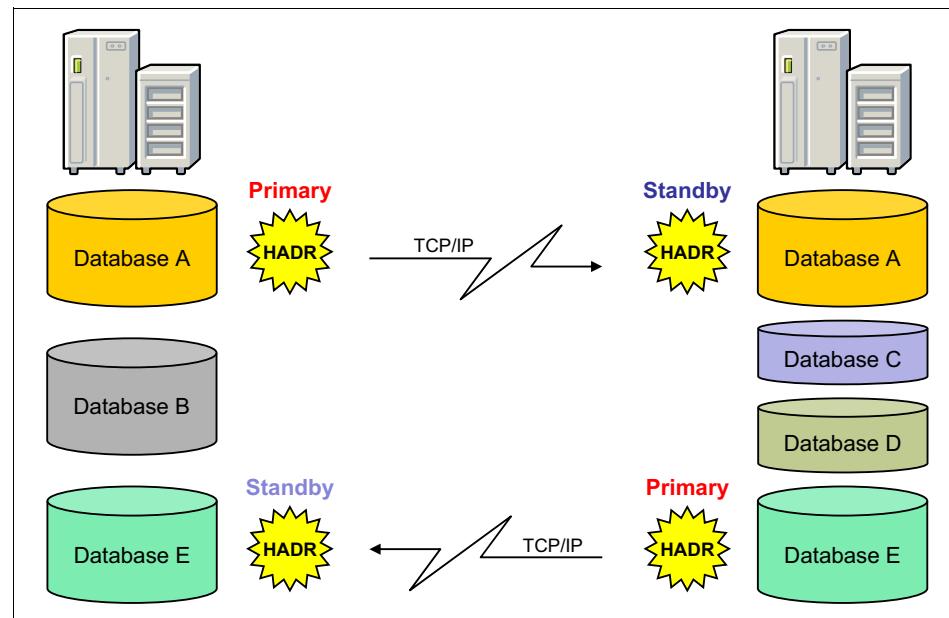


Figure 5-2 HADR takes place at the database level and not the DB2 instance level

Since DB2 10.1, the HADR feature also supports multiple standby databases. This feature enables an advanced topology where you deploy HADR in multiple standby mode. In this scenario, you can have up to three standby databases in your setup. You designate one of these databases as the *principal HADR standby database*; any other standby database is an *auxiliary HADR standby database*. As with the standard HADR deployment, both types of HADR standbys are synchronized with the HADR primary database through a direct TCP/IP connection. Furthermore, both types support the reads on standby feature, and you can configure both types for time-delayed log replay. In addition, you can issue a forced or non-forced takeover on any standby.

This scenario is illustrated by Figure 5-3.

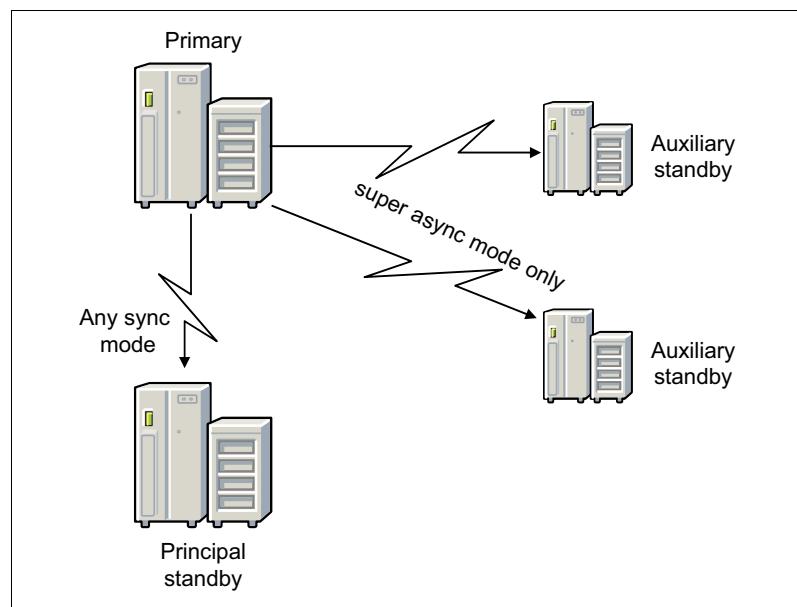


Figure 5-3 HADR scenario with multiple standby servers

There are many benefits to using a multiple HADR standby setup. Instead of employing the HADR feature to achieve your high availability objectives and another technology to achieve your disaster recovery objectives, you can use HADR for both. You can deploy your principal standby in the same location as the primary. If there is an outage on the primary, the principal standby can take over the primary role within your recovery time objectives (RTOs). You can also deploy auxiliary standbys in a distant location, which provides protection against a widespread disaster that affects both the primary and the principal standby.

When you work with multiple standby databases, all of the HADR sync modes are supported on the principal standby, but the auxiliary standbys can be only in SUPERASYNC mode.

Another point to consider while you set up an HADR environment is the new HADR delayed replay feature. This feature prevents data loss because of errant transactions. Delayed replay intentionally keeps the standby database at a point in time that is earlier than the point in time of the primary database by delaying replay of logs on that standby. If an errant transaction is run on the primary, you have until the configured time delay elapses to prevent the errant transaction from being replayed on the standby. To recover the lost data, you can either copy this data back to the primary, or you can have the standby take over as the new primary database.

You can use this feature in either single standby mode or multiple standby mode. In multiple standby mode, typically one or more standbys stays current with the primary for high availability or disaster recovery purposes, and one standby is configured with delayed replay for protection against errant transactions. If you use this feature in single standby mode, you should not enable IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) because the takeover fails.

5.1.2 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 are written to the primary's log files and after an acknowledgement is received that it is also written to the standby's log files. There can be no data loss in this mode if HADR is in the Peer state.
- ▶ **Near-synchronous**
This option is the default option. Log write is successful only when the primary's log buffer is written to log files on the primary and an acknowledgement is received that the log buffer is received on the standby.
- ▶ **Asynchronous**
Log write is successful when logs are written to the disk on the primary and log data is sent through TCP/IP to the standby. Data loss can occur in this mode.

- ▶ Super-asynchronous

The log writes are considered successful when the log records are written to the log files on the primary database. Because the primary database does not wait for acknowledgements from the standby database, transactions are considered committed regardless of the state of the replication of that transaction.

Figure 5-4 illustrates these HADR synchronization modes, and the point at which DB2 considers itself able to commit a transaction while in the Peer state.

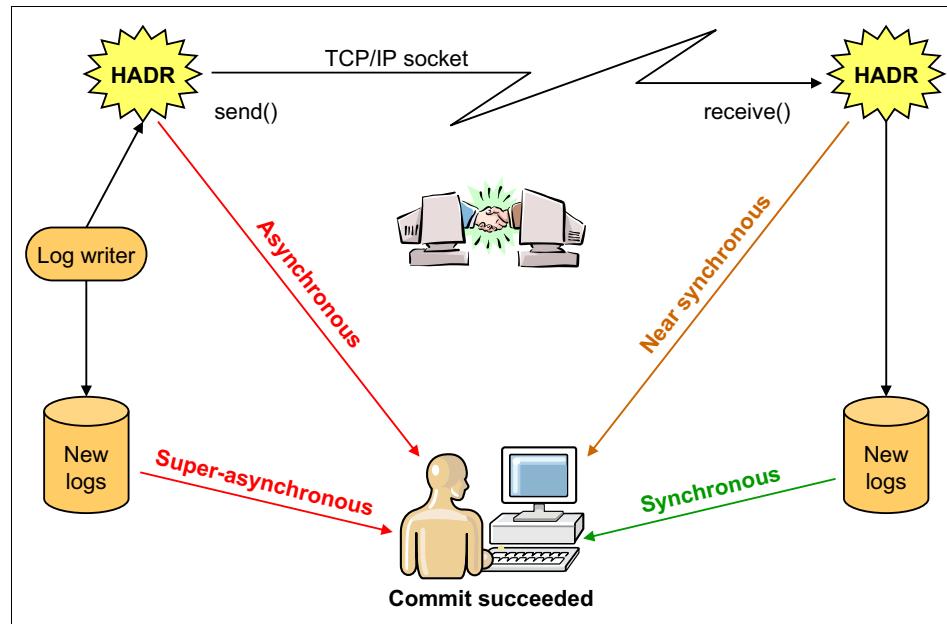


Figure 5-4 Synchronization modes

5.2 HADR architecture

HADR technology is tightly coupled to the DB2 logging and recovery strategy. The main design goal of DB2 HADR is to quickly fail over the failed system to a standby server if there is a failure on the primary server.

The basic architecture of HADR is simple. It is based on the shipping of log records from the primary database to a standby database. Thus, in DB2 the HADR implementation can be used by the database administrator (DBA) to decide which databases should be replicated.

To start HADR, the corresponding commands must be entered on both the standby and primary databases. The HADR system starts as soon as the primary database becomes active.

Regarding a HADR system, DB2 has two special engine dispatchable units (EDUs) to handle all HADR work. The one on the primary database is called *db2hadrp*. Its counterpart on the standby database is called *db2hadrs*. They are in charge of sending log records from the primary to the standby, processing system messages, and receiving log files on the standby system.

At the database activation of the primary, the *db2hadrp* is created; this thread then reads the database configuration file and opens a port to listen for a connection from the standby. The relevant HADR database configuration parameters are described in 11.1, “DB2 HADR configuration parameters” on page 400. 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. If the primary does not receive any communication from the standby database, then the primary concludes that the connection with the standby is lost. If the corresponding primary and standby are in the Peer state when the connection is lost, they move into the disconnected Peer state if the **hadr_peer_window** database configuration parameter is greater than zero, or into the remote catchup pending state if **hadr_peer_window** is not greater than zero.

If the **BY FORCE** clause is used when you run the **START HADR** command, the primary does not wait for the standby to connect. The primary creates a listener and listens on the HADR port. The standby initiates the connection to the primary. If the standby's connection attempt fails, it tries to connect to the primary again. 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 **BY FORCE** clause is that the database does not respect the setting of the **AUTORESTART** parameter and if a crash occurs, crash recovery is performed regardless 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 that is known as *heartbeats*. Heartbeats verify that both the primary and the standby can see each other. Heartbeats are special short messages that are 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 are 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 determined and the primary starts sending all log files that are required for the standby to reach the same sync point as the primary.

The db2lfr EDU captures changes that are made on the primary server either by reading the log buffer or by reading the log files. It then relays the log records to the db2hadrp thread, that in turn forwards the records to the db2hadrs thread on the standby. Finally, the db2hadrs thread passes the received log data to the local log system.

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

5.3 Terminology

Here we list and describe some of the more common terms that are 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 10.1 Information Center Glossary at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luwglossary.doc/doc/glossary.html>

Use 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 are 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.

- ▶ Cluster

A cluster is a system that is based on two or more nodes. Nodes are connected to each other and to shared storage. Each of these items is considered a cluster resource. A clustering software is used to logically connect these resources to appear as a single system to the clients.

▶ Cluster resource

An object that can be managed by a clustering software. Clustering software supports many resource types. These resource types include Disks, IP addresses, File Share, DB2 instance, and so on.

▶ Failover

In the context of HADR, this term refers to changing the status of the standby in an HADR pair to become the primary, with full DB2 functionality, because the original primary fails. The original primary can start then 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 the 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 that failed are made active on a previously passive or inactive node. This situation is nearly indistinguishable as a concept from failover in HADR.

▶ Failback

In the context of HADR, after a failover occurs, 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 because the failure of the new primary. This situation is especially likely in cases where the HA goal is ultra-fast failover, as failback can be viewed as a needless 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. This situation is especially true for 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 if there is a 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 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 components 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. 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* regarding the log stream. The usage 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 fail over DB2 functionality from a broken DB2 primary database to the standby. This period starts from the point at which failover is initiated to the point that DB2 functionality is restored. To a client that connects 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 records to the standby whenever it flushes a log record to the disk. The log records are replayed on the standby as they arrive. The records 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)

The principal (master) copy of the database. Applications apply updates to the primary database and those updates are propagated to the standby server through 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 join only as a standby. This concept is called primary reintegration. After reintegration, a failback can optionally be performed.

- ▶ Reads on standby feature

The reads on standby capability can be used to run read-only operations on the standby database in your HADR solution. You can use this expanded role of the standby database to use the standby in new ways, such as running some of the workload that would otherwise be running on your primary database.

- ▶ Standby (database)

A copy of the primary database. It is not updated directly by the application. All updates occur by rolling forward log data that is generated on the primary database.

- ▶ Auxiliary standby (database)

When you deploy the HADR feature in multiple standby mode, you can have up to three standby databases in your setup. You designate one of these databases as the principal HADR standby database; any other standby database is an auxiliary HADR standby database. The auxiliary standbys can be only in SUPERASYNC mode.

► Split brain

This term 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 command is a *manual* one 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 receives an acknowledgment message from the standby confirming that the relevant log data is 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 if the primary was in a 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 is 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.

- ▶ Super-asynchronous mode

In this mode, the HADR pair can never be in the Peer state or disconnected Peer state. The log writes are considered successful when the log records are written to the log files on the primary database. Because the primary database does not wait for acknowledgement from the standby database, transactions are considered committed regardless of the state of the replication of that transaction.

- ▶ Takeover

Takeover is the act of the HADR standby taking 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 acts unilaterally (with the risk of dual primary/split brain). Failover is a unilateral takeover. Fallback is a form of cooperative role switching performed after first reintegrating the repaired server as a standby. HADR *takeover* can be performed outside the context of failover and fallback. A user might want to switch HADR roles for rolling upgrade, which does not involve a failure at all.

- ▶ Engine dispatchable unit (EDU)

The DB2 database server must perform many different tasks, such as processing database application requests or ensuring that log records are written out to disk. Each task is typically performed by a separate engine dispatchable unit (EDU). The engine dispatchable units (EDUs) are implemented as threads on all platforms. DB2 agents are the most common type of EDU. These agents perform most of the SQL and XQuery processing on behalf of applications. Prefetchers and page cleaners are other common EDUs.



HADR setup

In this chapter, we describe the steps to set up a High Availability Disaster Recovery (HADR) environment using both the DB2 command line and IBM Data Studio V3.1.1. We cover the basic operations, including the **START HADR**, **STOP HADR**, and **TAKEOVER HADR** commands. We also provide some troubleshooting tips.

This chapter covers the following topics:

- ▶ Requirements for setting up HADR
- ▶ Setup and configuration
- ▶ Basic operation
- ▶ Troubleshooting

6.1 Requirements for setting up HADR

Before you set up HADR, you must know the requirements for a HADR setup. We provide a basic outline of what you need to get started (for more information, see *Data Recovery and High Availability Guide and Reference*, SC27-3870-00).

6.1.1 Requirements

To set up HADR, you must have the following requirements in place:

- ▶ HADR is a DB2 feature available in all DB2 editions except for DB2 Express-C. For the standby and auxiliary standby databases, DB2 licensing takes place according to usage as hot warm or cold standby. For more information, consult with an IBM Information Management marketing representative.
- ▶ The operating system on the primary and standby databases should be the same version, including the patches. You can violate this rule for a short time during a rolling upgrade, but take extreme caution.
- ▶ A TCP/IP interface must be available between the HADR host machines.
- ▶ The DB2 version and level must be identical on both the primary and the standby databases.
- ▶ The DB2 software for both the primary and the standby databases must be the same bit size (32-bit or 64-bit).
- ▶ Buffer pool sizes on the primary and the standbys should be the same. 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. If you are using the reads on standby feature, you must configure the buffer pool on the primary so that the active standby can accommodate log replay and read applications.
- ▶ The primary and standby databases must have the same database name, which means they must be in different instances.
- ▶ Table spaces must be identical on the primary and standby databases, including:
 - Table space type (DMS or SMS)
 - Table space size
 - Container path
 - Container size
 - Container file type (raw device or file system)
- ▶ The amount of space that is allocated for log files should also be the same on both the primary and standby databases.

6.1.2 Parameters

Use the following parameters to set up HADR:

- ▶ Use identical host computers for the HADR primary and standby database servers. They should be from the same vendor and have the same architecture.
- ▶ Use a high-speed and 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.

6.2 Setup and configuration

Here we perform the steps to set up HADR on an existing installation of DB2. Our lab example uses 64-bit DB2 10.1 for SUSE Linux Enterprise Server 11 on Intel (kernel 3.0.34-0.7). However, these steps are applicable to all supported environments.

6.2.1 Preparing the environment

To prepare a real-world, pre-change window environment to set up HADR, there are a few things that you can plan for so that you are ready beforehand. Unless you are still using circular logging, the only downtime that is required for setting up HADR is for updating database configuration parameters. This downtime can be avoided by scheduling the work in a regular maintenance window. There should be no pressure to minimize your change window, but avoid periods of heavy logging if you want to avoid a long wait for the standby to catch up before you reach the Peer state.

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

As basic components for this lab, you need at least two DB2 instances of the same fix pack level. If you plan to set up a HADR environment with multiple standby databases, you must have at least three DB2 instances of the same fix pack level.

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

Determine whether you intend to use the IBM Data Studio or the DB2 *command line processor* (CLP) to configure HADR. IBM Data Studio is perfect for introducing people who are unfamiliar with the HADR concept.

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 working, whether locally or remotely, such as AIX with telnet-only access, or systems without Java, or systems with limited administrative network bandwidth.

If you work in a high-security environment where people that are performing DB2 installations 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 the HADR setup from IBM Data Studio as the DB2 instance owner, this situation specifically requires that the DB2 instance owner have the authority to update the /etc/services file (%Systemroot%\system32\drivers\etc\services on Windows), unless these ports are 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 servers that you use in a real-world implementation must be connected by a reliable TCP/IP connection. The servers, or at least the main port for each DB2 instance, must not have a firewall between them, 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 with a slow connection (use SUPERASYNC)
- ▶ Over an intranet WAN (use ASYNC or NEARSYNC, but use them more for disaster recovery functionality instead of high availability)
- ▶ On a LAN (use NEARSYNC)
- ▶ On servers that are physically next to each other in the same rack (use SYNC)

Our example uses the last configuration.

If your environment does not tolerate frequent backups, even if they are online, then take whatever steps are necessary to get in a maintenance window to perform a backup as a starting point for the HADR change window.

If you want to reduce the number of steps that are performed inside IBM Data Studio, in addition to changing from circular to archival logging, catalog entries for the remote system on both the servers beforehand:

- ▶ Use TCPIP NODE for the DB2 instance.
- ▶ Use a DATABASE alias for the DB2 database.

Example 6-1 and Example 6-2 show the commands to catalog the database server and database.

Example 6-1 Commands to catalog remote entries on the primary server (NODE1)

```
db2 catalog tcpip node DB2_NODE2 remote x.x.x.x server 60004  
remote_instance db2inst1 system NODE2  
db2 catalog database SAMPLE as SAMPL_NODE2 at node DB2_NODE2
```

Example 6-2 Commands to catalog remote entries on the standby server (NODE2)

```
db2 catalog tcpip node DB2_NODE1 remote x.x.x.x server 60004  
remote_instance db2inst1 system NODE1  
db2 catalog database SAMPLE as SAMPL_NODE1 at node DB2_NODE1
```

For our lab example, we use HADR ports 55001 for NODE1 and 55002 for NODE2. You can also arrange to have the unqualified remote host name registered in the /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 IBM Data Studio to enter the physical port number for the local and remote DB2 instances rather than the service name (port 60004 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.

You are now ready to perform an HADR setup.

6.2.2 Configuration using the HADR setup wizard

Figure 6-1 illustrates the basic intended high-level architecture of our lab environment.

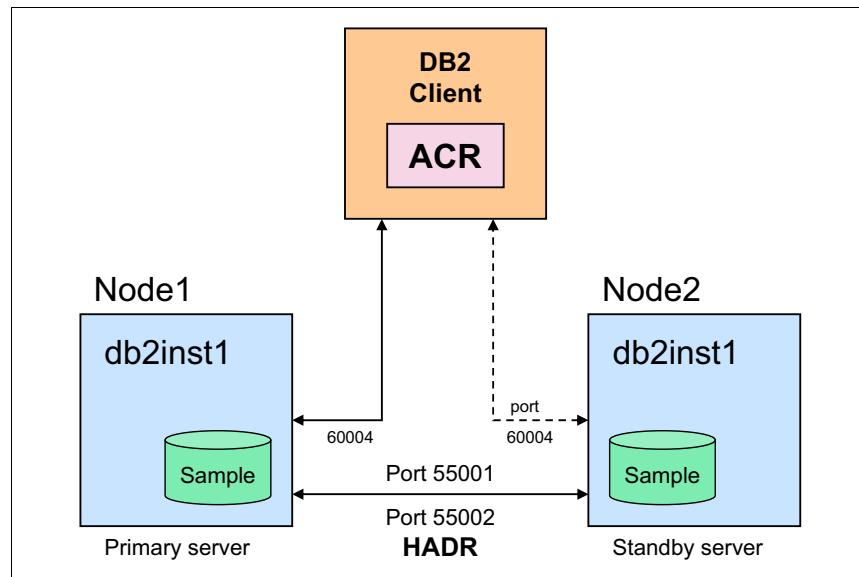


Figure 6-1 Lab environment high-level overview

Figure 6-1 shows two example instances both, called DB2INST1 on the server host names NODE1 (9.162.141.44) and NODE2 (9.162.141.48), in a non-clustered configuration, with basic TCP/IP Ethernet network adapter connectivity. NODE1 is initially set up as the HADR primary, and NODE2 is initially the HADR standby.

The ACR box in Figure 6-1 means *automatic client reroute*, which is built into the DB2 client and configured on the DB2 server. After it is configured, ACR is responsible for routing the client connections to whichever DB2 instance is set as the primary. For more information about ACR, see Chapter 10, “Automatic client reroute” on page 377.

In our lab, clients can communicate with the DB2 instance through port 60004, and DB2 HADR servers communicate with each other through ports 55001 and 55002.

The database that is set up in a HADR configuration is called SAMPLE; we catalog this database on each remote server, so that it is accessible from the Administration Explorer view in IBM Data Studio, and for general remote takeover, stop, and start commands.

In our lab, we start out with only one SAMPLE database, which is in the DB2INST1 instance on one of the servers (NODE1).

We finish with SAMPLE acting as the HADR primary on the DB2INST1 instance on NODE1, and as the HADR standby in the DB2INST1 instance on NODE2.

In the descriptions that follow, the term *remote* is used in the context of the server that we configure as the *standby* database, and the term *local* is used interchangeably with *primary*. We are running the setup locally on the server that becomes the primary. Initially, we do not have a primary or a standby, just the local server where IBM Data Studio is running (NODE1 for our example) and the remote server that has a database restored on it (NODE2).

After the configuration is complete, the primary and standby roles can be switched if you want, so these descriptions are temporary at best. To help with establishing a more permanent frame of reference, we included in our lab example the server names (NODE1 and NODE2) in parentheses when we listed a server, so you can substitute your own server names for each situation.

Beginning HADR configuration

For beginners, the preferred way to achieve a working HADR environment is to use IBM Data Studio. Complete the following steps:

1. Start IBM Data Studio, and expand the **All Databases** tree that is located in the Administration Explorer view to show your databases.
2. Connect to the database you want to set up HADR for by right-clicking the database in the Administration Explorer and select **Connect**. If prompted for connection parameters, enter your connection parameters and click **OK**.

3. Right-click the database that you want to set up and select **Set Up and Configure** → **Setup HADR...** (Figure 6-2).

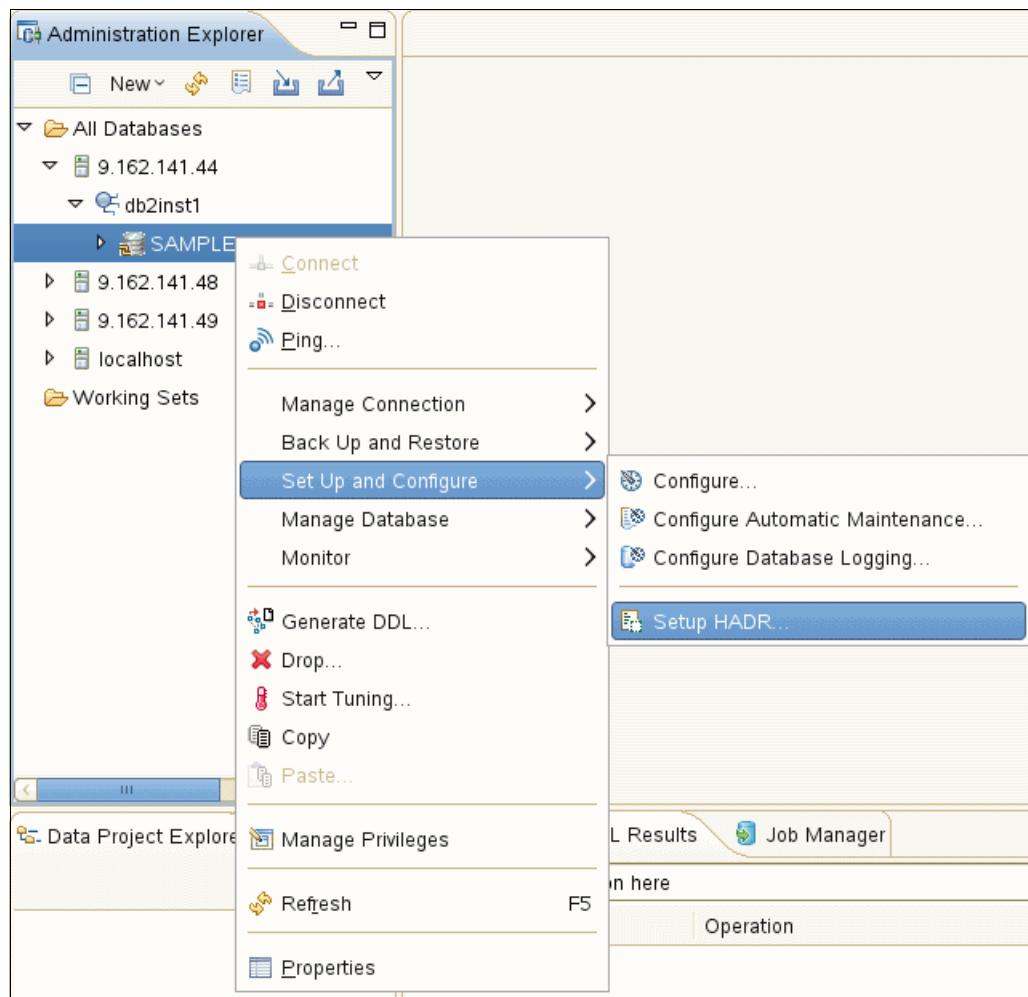


Figure 6-2 Starting the HADR Setup wizard

4. Check the logging mode of the primary database (Figure 6-3). This message points out that HADR requires the database to use archival logging.



Figure 6-3 Configure the primary database to use archival logging

If the database is still using circular logging, click **Configure...**, and complete the steps to configure archival logging.

If the database is already in archival logging mode, you can skip the next several substeps (steps a - j on page 182) and go straight to step 5 on page 182.

- a. If the database is in circular logging mode, the first window that opens after you click **Configure** is **Configure your logging type**. Select **Archive** as the database logging type.
- b. Choose how you would like to handle your archived logs: Click **Automatic DB2 archive** and specify a location, such as /usr/db2/arch1og (for Linux) or C:\db2arch1og (Windows) for the primary's archive log files. Click **Logging Size** in the right pane of the window.
- c. Choose the number and size of your log files: You can accept the defaults, as we do in our example, but if you have different requirements, see Chapter 5, “Databases”, in *DB2 10.1 Database Administration Concepts and Configuration Reference*, SC27-3871-00. Click **Logging Location**.
- d. Specify the location of your log files: Specify a location such as /usr/db2/act1og (Linux) or C:\db2act1og (Windows). Click **Backup Image**.
- e. Specify where to store your backup image: This action is required as part of the conversion to archival logging, as the database is placed in a backup pending state after you change the critical database configuration parameters. The logging recommences in the new archival format from that point in time. Here we specify a location such as /usr/db2/backup (Linux) or C:\db2backup (Windows). Click **Backup Options**.

- f. Specify options for the backup: Accept the defaults, and choose to compress the backup image if you want to save time to transfer the file to the standby server later on. Click **Preview Command** in the upper pane of the Settings window.
 - g. Review the actions that take place when you click **Run**: Here you can see a summary of the commands that are run when you click **Run**. Click **Run** or go back to one of the previous steps to fix any incorrect settings as necessary.
 - h. A warning message is displayed in the SQL Results window, warning you that the database is going to be deactivated and reactivated before the changes to one or more of the configuration parameters be effective.
 - i. The progress of the backup is indicated by the progress bar in the message section. Wait for it to complete.
 - j. The SQL Results pane in the lower section of the window gives you a summary of the results. Close the Configure Database Logging window.
5. Identify a standby database. After your database logging is in a state where you can start the HADR configuration and the configuration is deactivated, you can proceed with the next step which is choosing your standby database (Figure 6-4 on page 183). For this step, it is not relevant if you have a database on another server; have a DB2 instance running there.

Settings
Specify any additional settings to use. Click Run when you are done.

[Preview Command](#) [Run](#)

Primary Database

Standby Database

- [Copy Objects](#)
- [TCP/IP Parameters](#)
- [Client Reroute](#)
- [Synchronization](#)
- [Options](#)

Configure the standby database selection

Specify the connection profile for the standby database, how to initialize the standby database, and, if the standby database is new, the backup image of the primary database to restore the standby database.

-Standby database system

Select an existing connection profile for the standby database or create a new profile. The standby system must have the same operating system and database levels as the primary database.

Connection profile: [Add...](#)

Instance name:

-Standby database initialization options for connection profile

Use an existing database on the standby system
 Create a new database by using a backup image

-New standby database options

Select the backup image of the primary database to restore on the standby system. If the image file exists on the primary database system, the file is transferred before the HADR setup commands are run.

-Method for selecting a backup image

Enter the backup image information manually
 Select the backup image from a table

-Backup image

Use the following table to select the backup image to restore:

Time	Object Type	Table Space	Media Type	Location	Backup Type
07/12/2012 13:10:4	Database	5	File System	/home/db2inst1	FULL OFFLINE

-Standby system directory for backup image

Image copy location: [Browse...](#)

Overwrite the selected backup image if it already exists on the standby system

Figure 6-4 Choosing a database to use as a standby database

If you have not done so, catalog the standby server by providing the IP/host name, Instance name, and either specify a backup image of the primary server, or an existing remote database. To create a connection profile for your standby database, click **Add....** Add the host name/IP address and port number for the remote *system* and the name of the instance that is running on the standby system. You must provide a user name and password to connect to the remote instance; use any SYSADM user name from the remote instance (NODE2).

In the Add dialog box, the three fields for the details of the remote server (Host, Port, and Instance name) are described as follows:

- Host name: Specify the host name or TCP/IP address of the standby database server.
- Port number: Specify the TCP/IP connection port for the selected database on the host.
- Instance name: Specify the name of the DB2 instance that is running on the standby system.

For the field that specifies the port number, we used port 60004 for both DB2INST1 on the primary and DB2INST1 on the standby, that is, the DB2 port for the DB2 service for the instance on each server. This port is the one with which clients connect to DB2.

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

Figure 6-5 on page 185 shows the dialog box that prompts you for the details about the standby database. Enter the required information and click either **Test connection** to test the connection or **Finish** to create your connection profile.

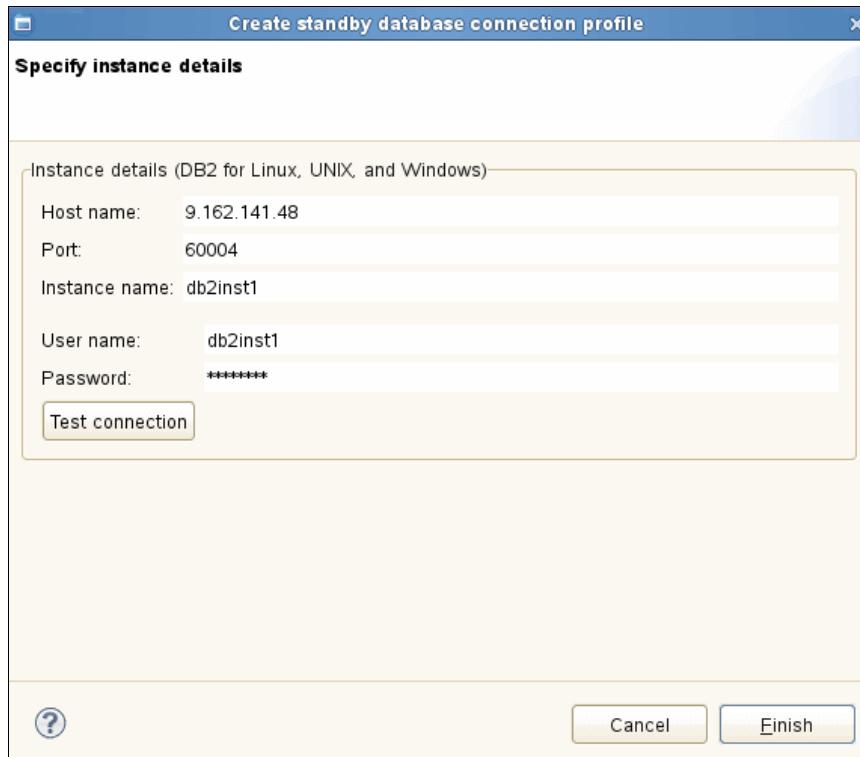


Figure 6-5 Catalog an instance node name for the remote server's DB2 instance

6. To set up HADR, you can use either of the two options that are shown in the Standby database initialization options for connection profile section in Figure 6-4 on page 183 to initialize the database on the standby system. You can either use an existing database or create a database on the standby system using a backup of the database that is located on the primary. The second one (Create a database by using a backup image) should provide a simple HADR setup, which is what we used in our example.
7. Specify a backup image of the primary database (see Figure 6-4 on page 183). If you previously set up your database for archival logging, you must have created a backup, which is perfect for a HADR configuration.
If you were already using archival logging, this point is when you must make a backup of your primary database. You can accomplish this task by selecting the **Primary Database** section and clicking **Backup**. After the backup is complete, you can use this backup to restore the database on the standby database.

In the Configure standby database window, you have two methods you can use to restore the database on the standby database. In the Method for selecting a backup image pane, you can either select **Enter the backup image information manually** or **Select the image from a table**. We choose to click select the backup image from a table as shown in Figure 6-4 on page 183.

After you make your choice, a pane opens where you can select one of the previously made backup images. Because we made a backup while we changed the database logging from circular to archival mode, we choose the latest backup in the Backup image section of the window.

IBM Data Studio is able to use DB2 ports to transfer the backup image on the local system to the remote system. To accomplish this task, you must provide a path on the standby where the backup image is to be copied to. You can accomplish this task by using the Standby system directory for backup image pane, which is the last pane in the window. Specify a path where the backup image should be copied to.

If the backup image that you intend to restore from is not listed in the table (for example, if the backup image is being transferred to the remote server over FTP or through file sharing), then select **Enter the backup image information manually** and specify the subdirectory and the date and time for the backup image you are using, as found in the backup file name. Click **Copy Objects** to proceed.

- Copy objects to the standby system (Figure 6-6). If you have DB2 objects that are not copied inside a database backup file, such as external code for UDFs and stored procedures, you can identify those objects in this window and have DB2 move them for you. Because we have none of those objects in our example, we leave the window blank and click **TCP/IP Parameters** to proceed.

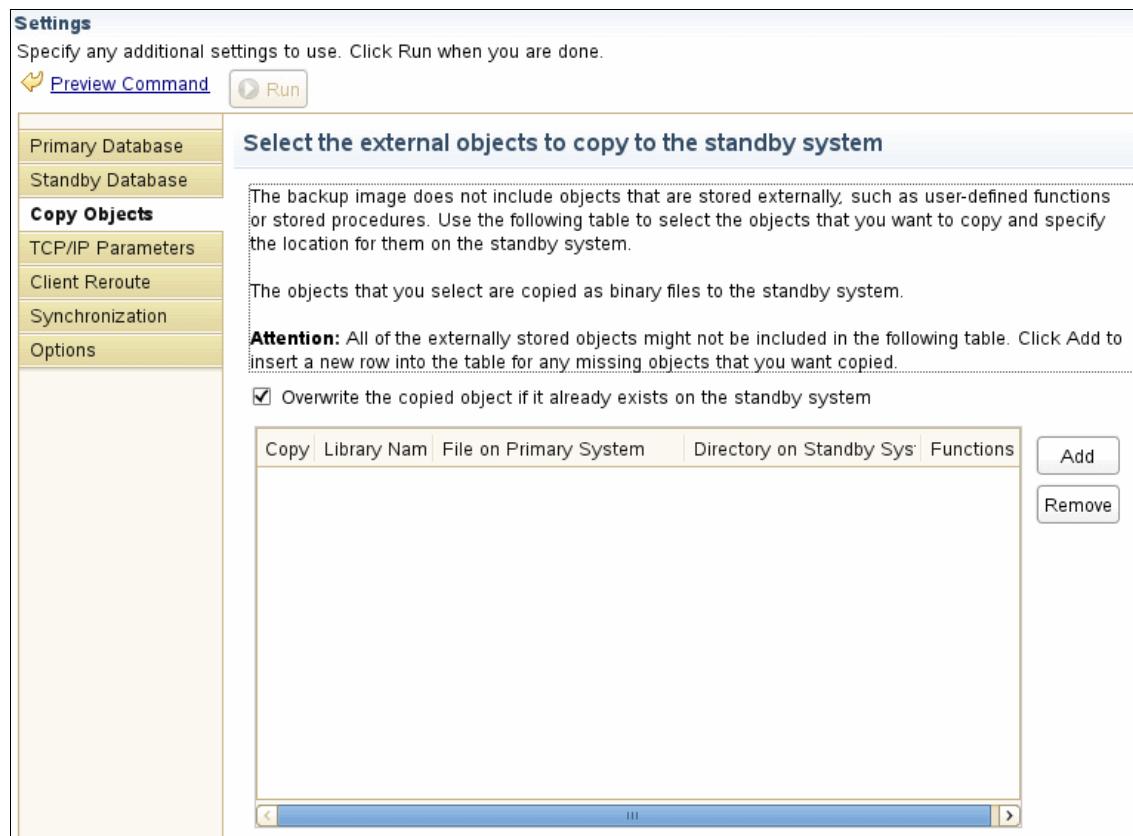


Figure 6-6 Copy “special” objects across to the standby system

9. Specify TCP/IP communication parameters. Figure 6-7 shows the settings that are required for the HADR-specific host name and service/port number for the primary and standby databases. The HADR feature establishes a connection between the primary and the standby databases. The TCP/IP parameters (service name/port number and host name) for both databases are retrieved from the database configuration information. Enter unique values (as required) and click **Client Reroute** to proceed.

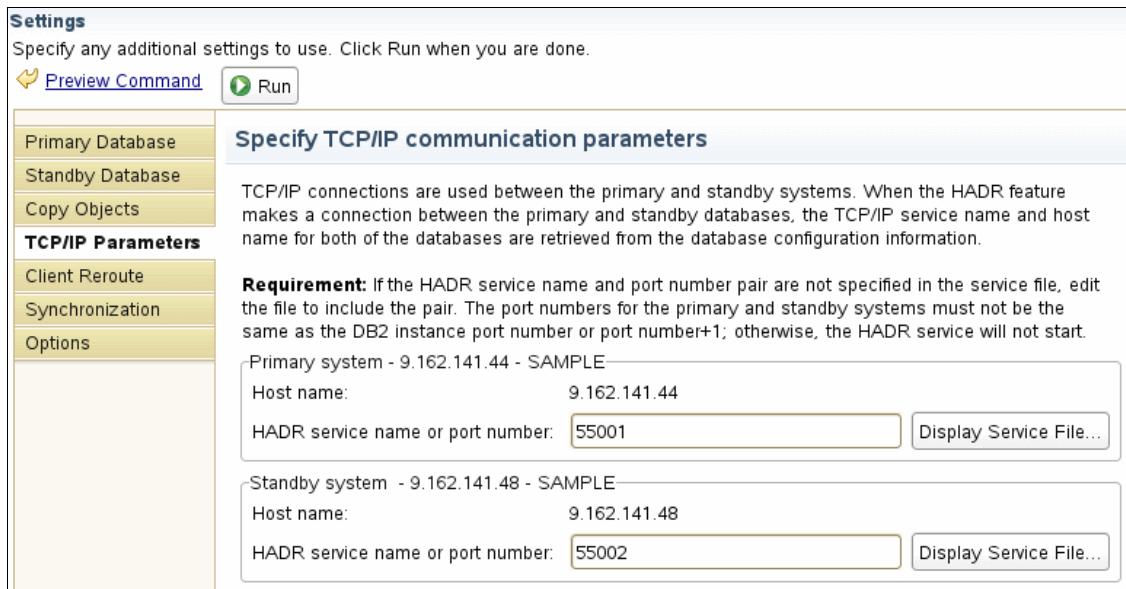


Figure 6-7 *Specify host address and port numbers for both HADR communication channels*

10. Configure databases for automatic client reroute. Figure 6-8 shows where we set up the IP address redirect facility. Each client connection to a DB2 server stores the IP address and port number of the primary and standby server. When you switch roles, or force a takeover, the other IP address is then used by the client; all a client sees is an SQL message that informs him that the connection is changed and the transaction or database connection should be tried again. For more information about ACR, see Chapter 10, “Automatic client reroute” on page 377.

The port number refers to the main port number for the DB2 instance, not to the HADR port. 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 **Synchronization** to proceed.

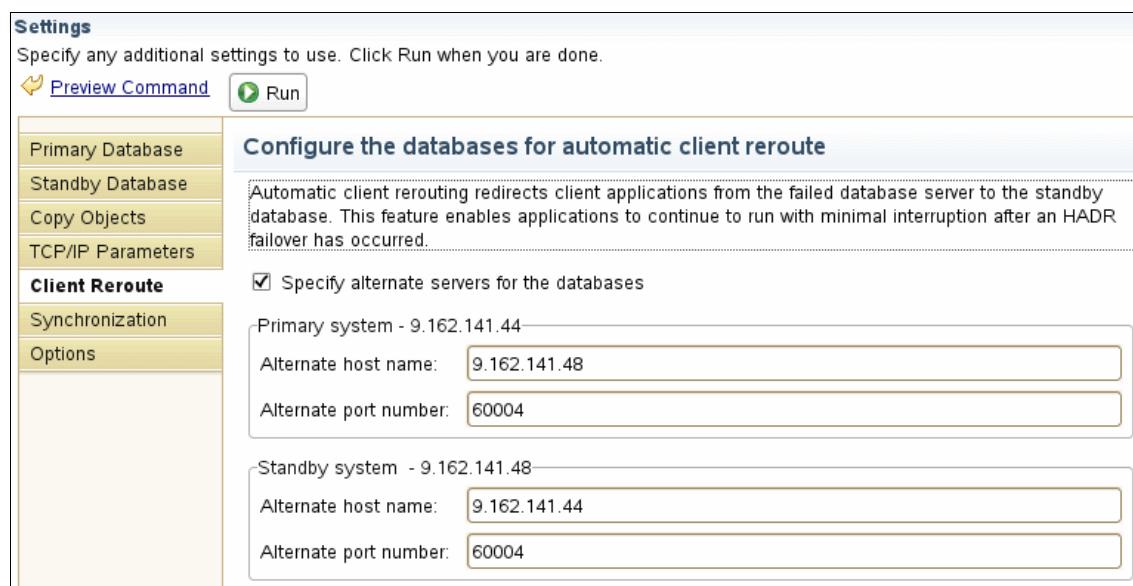


Figure 6-8 Configuring the automatic client reroute server and port information

11. Specify synchronization mode for Peer state log writing. In Figure 6-9, you can choose whether you want synchronous, near-synchronous, asynchronous, or super-asynchronous mode for HADR. In our example, we choose synchronous. For more information about the four available synchronization modes, see 5.2, “HADR architecture” on page 165. Specify values for the time-out period and the peer window time length. In our example, we accept the default values. For details about choosing the synchronization mode and how to speed up takeover time, see Chapter 11, “HADR configuration parameters and registry variables” on page 399. Click **Options** to proceed.

The screenshot shows a configuration dialog for HADR synchronization mode. The left sidebar has tabs for 'Primary Database', 'Standby Database', 'Copy Objects', 'TCP/IP Parameters', 'Client Reroute', 'Synchronization', and 'Options'. The 'Synchronization' tab is selected. At the top right are 'Preview Command' and 'Run' buttons. The main area is titled 'Specify synchronization mode for peer state log writing'. It contains a detailed description of the synchronization mode, which applies only when primary and standby systems are in peer state. It lists three options: Synchronous (selected), Near synchronous, and Asynchronous. Below each option is a brief description. Under 'Connection time-out period', the value '120' is entered in the input field. Under 'Peer window time length', the value '0' is entered in the input field. Both fields have '(seconds)' suffixes.

Settings
Specify any additional settings to use. Click Run when you are done.

[Preview Command](#) [Run](#)

Synchronization mode

- Synchronous
This mode guarantees that no data will be lost. Failback is supported.
- Near synchronous
This mode guarantees no loss of data for single point-of-failure disasters. If no loss of data has occurred, failback is supported.
- Asynchronous
In this mode, the primary system does not wait for log data to reach the standby system, and log data is likely to be lost in the event of a failover. If HADR detects a loss of log data, failback is not supported.

Connection time-out period
Time-out period (1 - 4294967295): (seconds)

This parameter (hadr_timeout) specifies the time that the high availability disaster recovery (HADR) process waits before considering that a communication attempt has failed and the connection to the standby database is lost. Set the parameter to a value that is long enough to avoid false alarms on the HADR connection caused by short, temporary networks interruptions.

Peer window time length
Peer window time length (0 - 4294967295): (seconds)

This parameter (hadr_peer_window) specifies the length of time that a HADR primary-standby pair continues to be considered in peer state after a connection to the standby database is lost. The standby database is placed in disconnected peer state for the length of time specified. Set the parameter to a value that is long enough to allow the system to perform automated failure response.

Figure 6-9 Choose an HADR syncmode - SYNC, NEARSYNC, or ASYNC

12. Choose the Setup HADR options. Figure 6-10 shows the HADR options window. Here you can choose to either start the HADR services immediately after the setup is complete, or to start them later on your own. For our example, we choose to start the HADR services immediately. Furthermore, you can set some registry variables. For example, you can enable the standby database to accept read-only workloads.

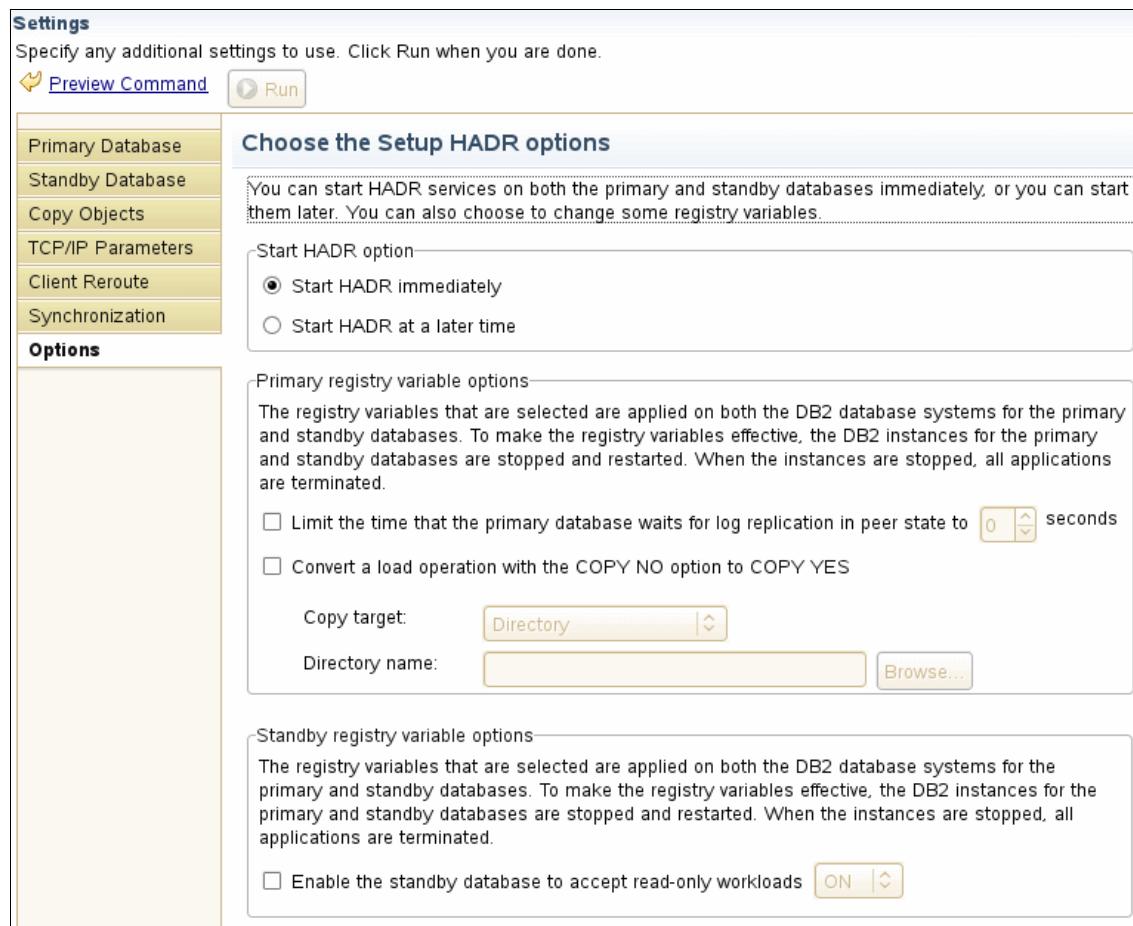
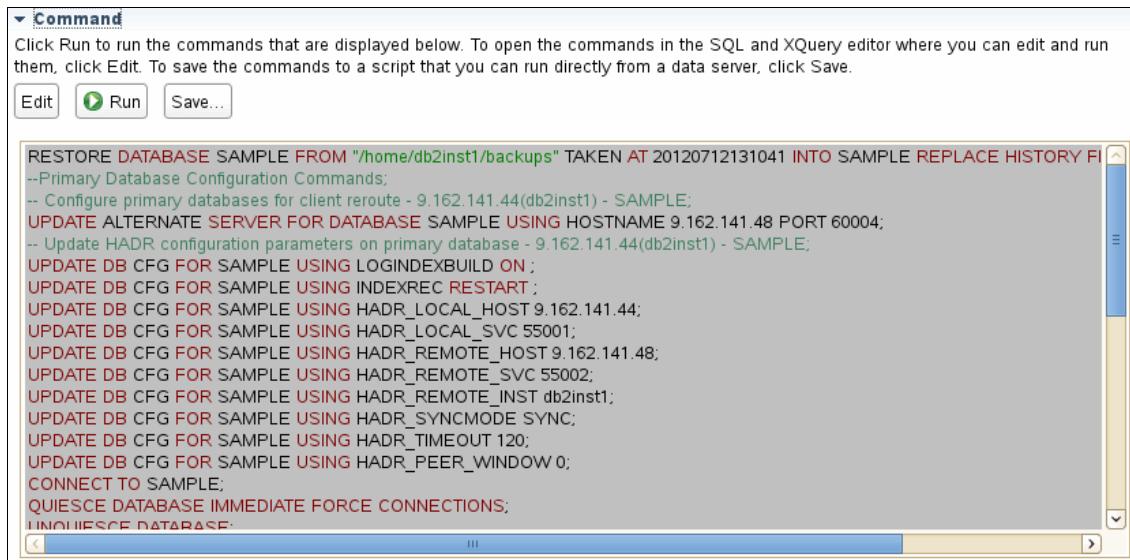


Figure 6-10 Choose option parameters

13. Review the actions that take place when you click **Run** (Figure 6-11). By clicking the **Preview Command** link, you can review the command-line statements that are run when you hit the **Run** button. These commands are worth saving in a text file for subsequent configurations. To start the setup process, click **Run**.



The screenshot shows a window titled 'Command' with a sub-section 'Preview Command'. It contains a text area with several SQL commands related to HADR setup. At the top of the window, there are three buttons: 'Edit', 'Run' (which is highlighted with a green border), and 'Save...'. Below the buttons, a note says: 'Click Run to run the commands that are displayed below. To open the commands in the SQL and XQuery editor where you can edit and run them, click Edit. To save the commands to a script that you can run directly from a data server, click Save.' The text area displays the following commands:

```
RESTORE DATABASE SAMPLE FROM "/home/db2inst1/backups" TAKEN AT 20120712131041 INTO SAMPLE REPLACE HISTORY FILE;
--Primary Database Configuration Commands;
-- Configure primary databases for client reroute - 9.162.141.44(db2inst1) - SAMPLE;
UPDATE ALTERNATE SERVER FOR DATABASE SAMPLE USING HOSTNAME 9.162.141.48 PORT 60004;
-- Update HADR configuration parameters on primary database - 9.162.141.44(db2inst1) - SAMPLE;
UPDATE DB CFG FOR SAMPLE USING LOGINDEXBUILD ON ;
UPDATE DB CFG FOR SAMPLE USING INDEXREORG RESTART ;
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_HOST 9.162.141.44;
UPDATE DB CFG FOR SAMPLE USING HADR_LOCAL_SVC 55001;
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_HOST 9.162.141.48;
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_SVC 55002;
UPDATE DB CFG FOR SAMPLE USING HADR_REMOTE_INST db2inst1;
UPDATE DB CFG FOR SAMPLE USING HADR_SYNCMODE SYNC;
UPDATE DB CFG FOR SAMPLE USING HADR_TIMEOUT 120;
UPDATE DB CFG FOR SAMPLE USING HADR_PEER_WINDOW 0;
CONNECT TO SAMPLE;
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
```

Figure 6-11 Reviewing the commands before you start the final HADR setup

14. Execution of steps. After you click **Run**, the steps of the previously generated script are run. Figure 6-11 shows a summary of those commands. If any issues are encountered, IBM Data Studio informs you about what went wrong and what must be corrected to successfully complete the corresponding step.

Figure 6-12 shows the completed step, which is what you see when the script runs successfully. The standard HADR setup with one primary and one standby is now complete. Close the HADR setup window to complete the setup process.

The screenshot shows the IBM Data Studio interface with the 'SQL Results' tab selected. In the main pane, there is a table titled 'Status' with two columns: 'Status' and 'Operation'. The table contains six rows, each representing a step in the HADR setup process. The steps are:

Status	Operation
Warning	Configure Database Logging SAMPLE
Succeeded	BACKUP DATABASE SAMPLE TO "/ho
Succeeded	RESTORE DATABASE SAMPLE FROM
Succeeded	Setup HADR SAMPLE
Succeeded	Setup HADR SAMPLE
Succeeded	Setup HADR SAMPLE

Below the table, the status bar displays: 'Displayed 6 of 6 results: 5 succeeded, 0 failed, 0 terminated, 1 warning, 0 critical error'. To the right of the table, the 'Status' panel shows the command: 'Setup HADR SAMPLE'. The output pane contains the following text:

```
-- Start HADR on primary database - 9.162.141.44(db2inst1) - SAMPLI  
DEACTIVATE DATABASE SAMPLE  
START HADR ON DATABASE SAMPLE AS PRIMARY
```

The status bar also shows: 'Transferring backup ima...system ...'

Figure 6-12 SQL Results window - the completed steps for a scripted HADR setup

15. Verify the HADR setup. You can check if your HADR setup is working correctly by using IBM Data Studio.

Right-click the corresponding database in the Administration Explorer and select **Manage Database** → **Manage HADR**. A window similar to the one shown in Figure 6-13 opens.

Among other things, you can see here that the databases are running in the Peer state, with the appropriate synchronization mode, and that they are in a connected state. In addition, you can also see the log information. You can get similar information by using the **MON_GET_HADR** table function or the **db2pd** command.

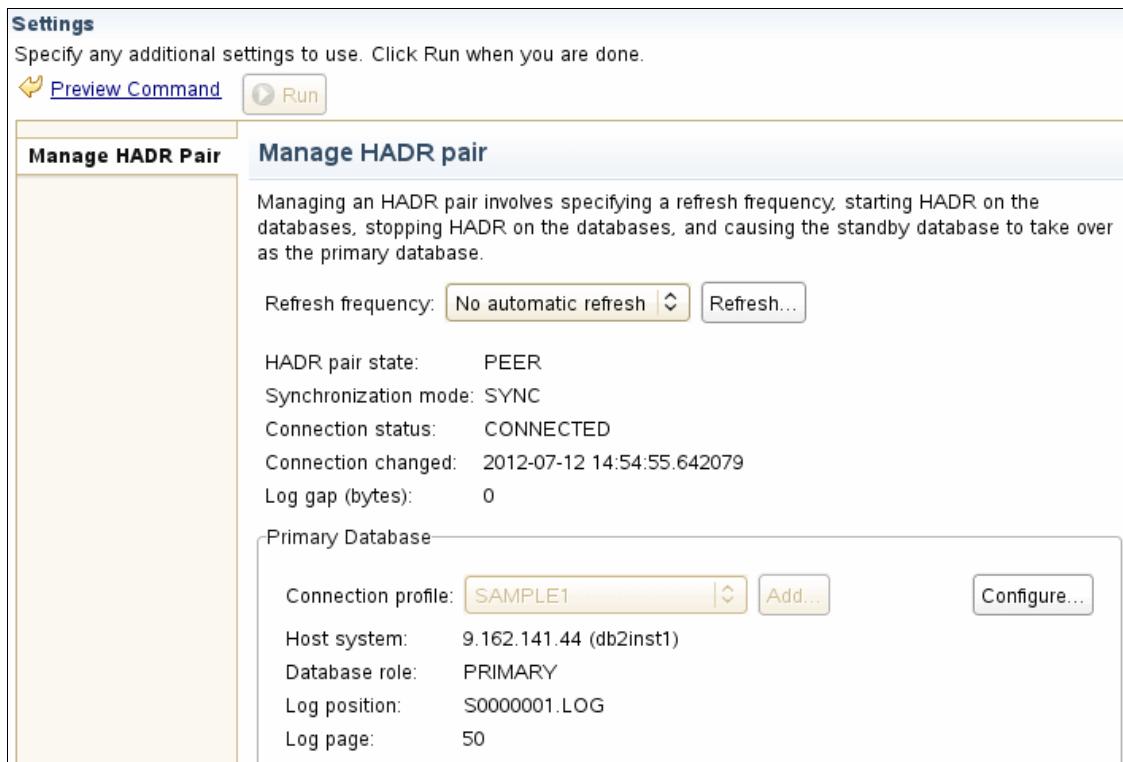


Figure 6-13 IBM Data Studio - manage the HADR window

6.2.3 Command-line setup

DB2 provides both a command line and a GUI setup for HADR. In this section, we demonstrate how to set up HADR using the command-line interface (CLI). In our lab environment, we set up the NODE1 server as the primary server, and NODE2 server as the standby server with the database SAMPLE on both servers.

To set up HADR through the CLI, complete the following steps:

1. Set the required database configuration parameters.

If archive logging is not already turned on, then update the **LOGARCHMETH1** parameter by running the following command:

```
db2 update database configuration for sample using LOGARCHMETH1  
disk:/dbuser/archived_logs
```

Also, set the **LOGINDEXBUILD** parameter so that index creation, recreation, or reorganization operations are logged, by running the following command:

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

2. Back up your database on the primary by running the following command:

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

3. Move a copy of the backup to the standby server.

4. Restore the database on your standby server.

In our example, we restore the SAMPLE database on the NODE2 server by running the following command:

```
db2 restore database sample from /usr/db2/backup taken at  
20120716101759 replace history file
```

5. Configure databases for ACR. This step is optional, but is a good idea. To configure ACR, update the **ALTERNATE SERVER** database configuration parameter on both the primary and the standby server by completing the following steps:

- a. On the primary server, NODE1 in our test case, set the standby server as the alternate server by running the following command:

```
db2 update alternate server for database sample using hostname  
NODE2 port 60004
```

- b. On the standby server, NODE2 in our test case, set the primary server as the alternate server by running the following command:

```
db2 update alternate server for database sample using hostname  
NODE1 port 60004
```

6. Update the following fields in the services file on the primary (NODE1) and standby (NODE 2) server 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 (NODE1) by running the following commands:
 - `db2 update db cfg for SAMPLE using HADR_LOCAL_HOST NODE1`
 - `db2 update db cfg for SAMPLE using HADR_LOCAL_SVC DB2_HADR_1`
 - `db2 update db cfg for SAMPLE using HADR_REMOTE_HOST NODE2`
 - `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, NODE2 in our test case, by running the following commands:
 - `db2 update db cfg for SAMPLE using HADR_LOCAL_HOST NODE2`
 - `db2 update db cfg for SAMPLE using HADR_LOCAL_SVC DB2_HADR_2`
 - `db2 update db cfg for SAMPLE using HADR_REMOTE_HOST NODE1`
 - `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. If you want to use the reads on standby feature, you must set the corresponding registry variables on the standby server by running the following commands (this step is optional):
 - `db2set DB2_HADR_ROS=ON`
 - `db2set DB2_STANDBY_ISO=UR`
10. Start the standby database first by running the following commands on NODE2:
 - `db2 deactivate database SAMPLE`
 - `db2 start hadr on database SAMPLE as standby`

11. Start HADR on the primary database on NODE1 by running the following commands:

- `db2 deactivate database SAMPLE`
- `db2 start hadr on database SAMPLE as primary`

6.2.4 Setting up HADR with multiple standby servers

This section describes how to initialize HADR in multiple standby mode.

Initializing a HADR system in multiple standby mode is similar to single standby mode. The main difference is that multiple standby mode requires the `hadr_target_list` configuration parameter to be set on all participating databases.

This parameter lists the standby servers in a scenario where the database becomes a primary. It is required even on a standby. Mutual inclusion is required (that is, if server A has server B in its target list, B must have A in its target list). This setup ensures that after a takeover from any standby, the new primary can always keep the old primary as its standby. The first standby that you specify in the target list is designated as the *principal standby database*. Additional standbys are *auxiliary standby databases*. Working out the target list for each database is an important step.

To initialize HADR in multiple standby mode, complete the following steps:

1. Set the required database configuration parameters.

If archival logging is not already activated, update the `LOGARCHMETH1` parameter by running the following command:

```
db2 update db cfg for sample using LOGARCHMETH1  
disk:/dbuser/archived_logs
```

Also, set the `LOGINDEXBUILD` parameter so that index creation, recreation, or reorganization operations are logged, by running the following command:

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

2. Back up your database on the primary by running the following command:

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

3. Move a copy of the backup to each of the standby servers.

4. Restore the database on each of the standby servers.

In our example, we restore the SAMPLE database on NODE2 (the principal standby) and on NODE3 (the auxiliary standby) by running the following command:

```
db2 restore database sample from /usr/db2/backup taken at  
20120716153706 replace history file
```

5. Configure the databases for ACR. This step is optional, but is a good idea. Because each database server can have only one alternate server that is defined, you must select one standby database (usually the principal standby) as the alternate server of the primary. To configure ACR, update the **ALTERNATE SERVER** database configuration parameter on both the primary and each standby by completing the following steps:

- a. On the primary server, NODE1 in our test case, set the standby server as the alternate server by running the following command:

```
db2 update alternate server for database sample using hostname  
NODE2 port 60004
```

- b. On the principal standby server, NODE2 in our test case, set the primary server as the alternate server by running the following command:

```
db2 update alternate server for database sample using hostname  
NODE1 port 60004
```

- c. On the auxiliary standby server, NODE3 in our test case, set the primary server as the alternate server by running the following command:

```
db2 update alternate server for database sample using hostname  
NODE1 port 60004
```

6. Add the following fields in the services file on the primary (NODE1) and each standby (NODE 2, NODE3) server to set up the HADR communication:

- Service name: DB2_HADR_1
- Port number: 55001
- Service name: DB2_HADR_2
- Port number: 55002
- Service name: DB2_HADR_3
- Port number: 55003

7. Update the HADR database configuration parameters on the primary database (NODE1) by running the following commands:

- **db2 update db cfg for SAMPLE using HADR_LOCAL_HOST NODE1**
- **db2 update db cfg for SAMPLE using HADR_LOCAL_SVC DB2_HADR_1**
- **db2 update db cfg for SAMPLE using HADR_REMOTE_HOST NODE2**
- **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_TARGET_LIST NODE2:55002|NODE3:55003
 - 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 principal standby database, NODE2 in our test case, by running the following commands:
 - db2 update db cfg for SAMPLE using HADR_LOCAL_HOST NODE2
 - db2 update db cfg for SAMPLE using HADR_LOCAL_SVC DB2_HADR_2
 - db2 update db cfg for SAMPLE using HADR_REMOTE_HOST NODE1
 - 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_TARGET_LIST NODE1:55001|NODE3:55003
 - 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. Update the HADR database configuration parameters on the auxiliary standby database, NODE3 in our test case, by running the following commands:
 - db2 update db cfg for SAMPLE using HADR_LOCAL_HOST NODE3
 - db2 update db cfg for SAMPLE using HADR_LOCAL_SVC DB2_HADR_3
 - db2 update db cfg for SAMPLE using HADR_REMOTE_HOST NODE1
 - 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_TARGET_LIST NODE1:55001|NODE2:55002
 - db2 update db cfg for SAMPLE using HADR_SYNCMODE SUPERASYNC

- **db2 update db cfg for SAMPLE using HADR_TIMEOUT 3**
- **db2 update db cfg for SAMPLE using HADR_PEER_WINDOW 120**

10. If you want to use the reads on standby feature, you must set the corresponding registry variables on each standby by running the following commands (this step is optional):

- **db2set DB2_HADR_ROS=ON**
- **db2set DB2_STANDBY_ISO=UR**

11. Start the standby databases first by running the following commands on NODE2 and NODE3:

- **db2 deactivate database SAMPLE**
- **db2 start hadr on database SAMPLE as standby**

12. Start HADR on the primary database on NODE1 by running the following commands:

- **db2 deactivate database SAMPLE**
- **db2 start hadr on database SAMPLE as primary**

13. To verify that HADR is running, you can query the status of the databases from the primary on NODE1 by running the **db2pd** command, which returns all information about the HADR setup, including the standbys.

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

The standby databases start in the local catchup state, in which locally available log files are read and replayed. After all local logs are replayed, the databases enter the remote catchup pending state. After the primary starts, the standbys enter remote catchup state, in which log pages are received from the primary and replayed. After all of the log files that are on the disk of the primary database are replayed on the standbys, what happens next depends on the type of synchronization mode. A principal standby in SUPERASYNC and any auxiliary standby stay in remote catchup mode. A principal standby with a SYNC, NEARSYNC, or ASYNC mode enter peer mode.

The previous steps describe how to set up a HADR scenario with multiple standbys from scratch. If you have a working HADR environment (consisting of one primary and one standby) and want to add an additional auxiliary standby, you can find more information in the Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/t0060257.html>

6.2.5 HADR log spooling

The HADR log spooling feature is a new feature of DB2 10.1. It allows transactions on a primary to progress without having to wait for the log replay on the standby. When this feature is enabled, log data that is sent by the primary is spooled, or written, to disk on the standby, and that log data is later read by log replay.

The log spooling feature represents an improvement to the HADR feature. When replay is slow, it is possible that new transactions on the primary can be blocked because it is not able to send log data to the standby system if there is no room in the buffer to receive the data. The log spooling feature means that the standby is not limited by the size of its buffer. When there is an increase in data received that cannot be contained in the buffer, the log replay reads the data from disk. This situation allows the system to better tolerate either a spike in transaction volume on the primary, or a slow down of log replay (because of the replay of particular type of log records) on the standby.

This feature could potentially lead to a larger gap between the log position on the primary and the log replay on standby, which can lead to longer takeover time. You should consider your spool limit setting carefully, because the standby cannot start as the new primary and receive transactions until the replay of the spooled logs finishes.

The log spooling feature is activated by setting the `hadr_spool_limit` database configuration parameter.

```
db2 update db cfg for sample using HADR_SPOOL_LIMIT -1
```

The default value of 0 means no spooling. A value of -1 means unlimited spooling (as much as supported by the disk space available). If you are using a high value for `hadr_spool_limit`, and if there is a large gap between the log position of the primary and log replay on the standby, this setup might lead to a longer takeover time. The standby cannot assume the role of the new standby until the replay of the spooled logs finishes.

6.3 Basic operation

This section gives you the basic details about the start, stop, and takeover operations that are preformed for HADR. We give examples for both the command-line environment and IBM Data Studio.

6.3.1 Starting and shutting down

This section covers the startup and shutdown procedures for HADR in IBM Data Studio and the command-line environment.

Startup

Before you start HADR, ensure that your database manager (instance) on both the primary and standby databases is started. Run `db2start` to start the instance one each side, primary and standby. The instances can be started in any order.

When you start HADR, 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. Otherwise, the startup process fails to prevent a split-brain scenario.

Running the start command

Here is the syntax for the start command:

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

When you start the primary, the **BY FORCE** option specifies that the HADR primary database does not wait for the standby database to connect to it. After you run **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, run 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 details for the **START HADR** command in *DB2 10.1 Command Reference*, SC27-3868-00.

Starting HADR from IBM Data Studio

HADR can also be started from the Manage HADR window in the IBM Data Studio by completing the following steps:

1. From IBM Data Studio, expand the **All Databases** tree in the Administration Explorer down to the wanted database object. Right-click the database that you plan to start HADR on and select **Manage Database → Manage HADR** (Figure 6-14).

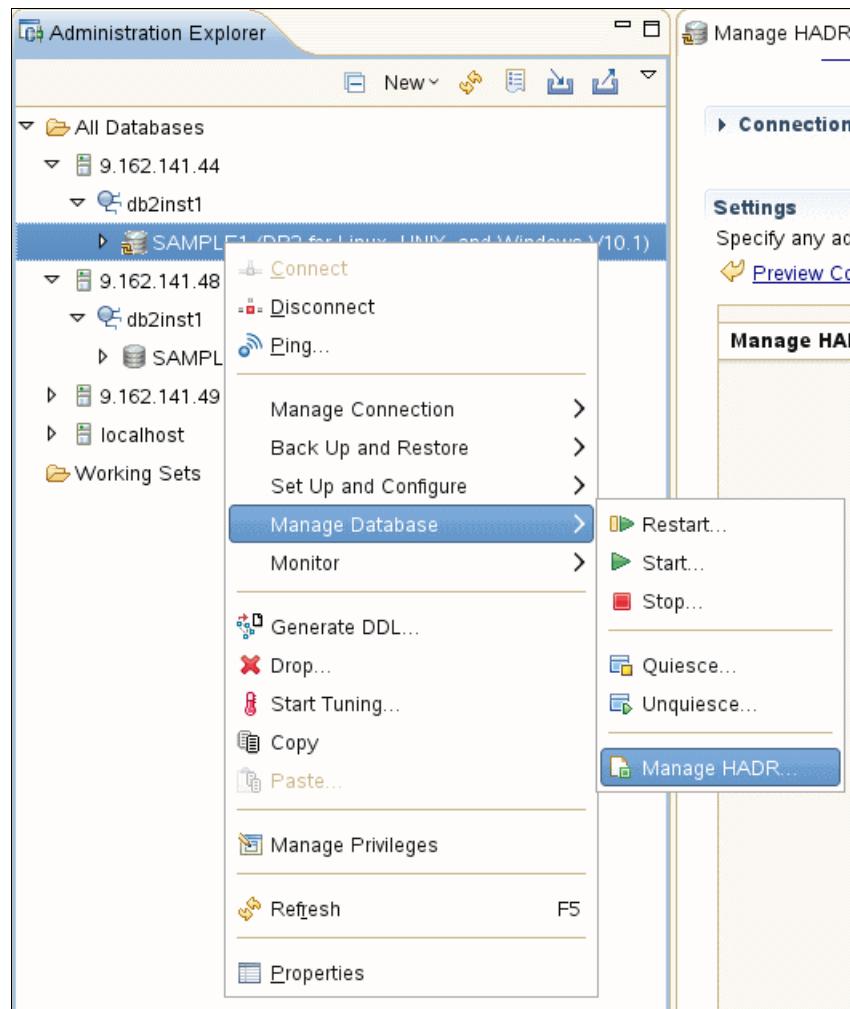


Figure 6-14 Select Manage HADR

- From the Manage High Availability Disaster Recovery (HADR) window, check the **Manage primary/standby database** check box and select the **Start HADR service** option in each pane (Figure 6-15). Click **Run** to start HADR.

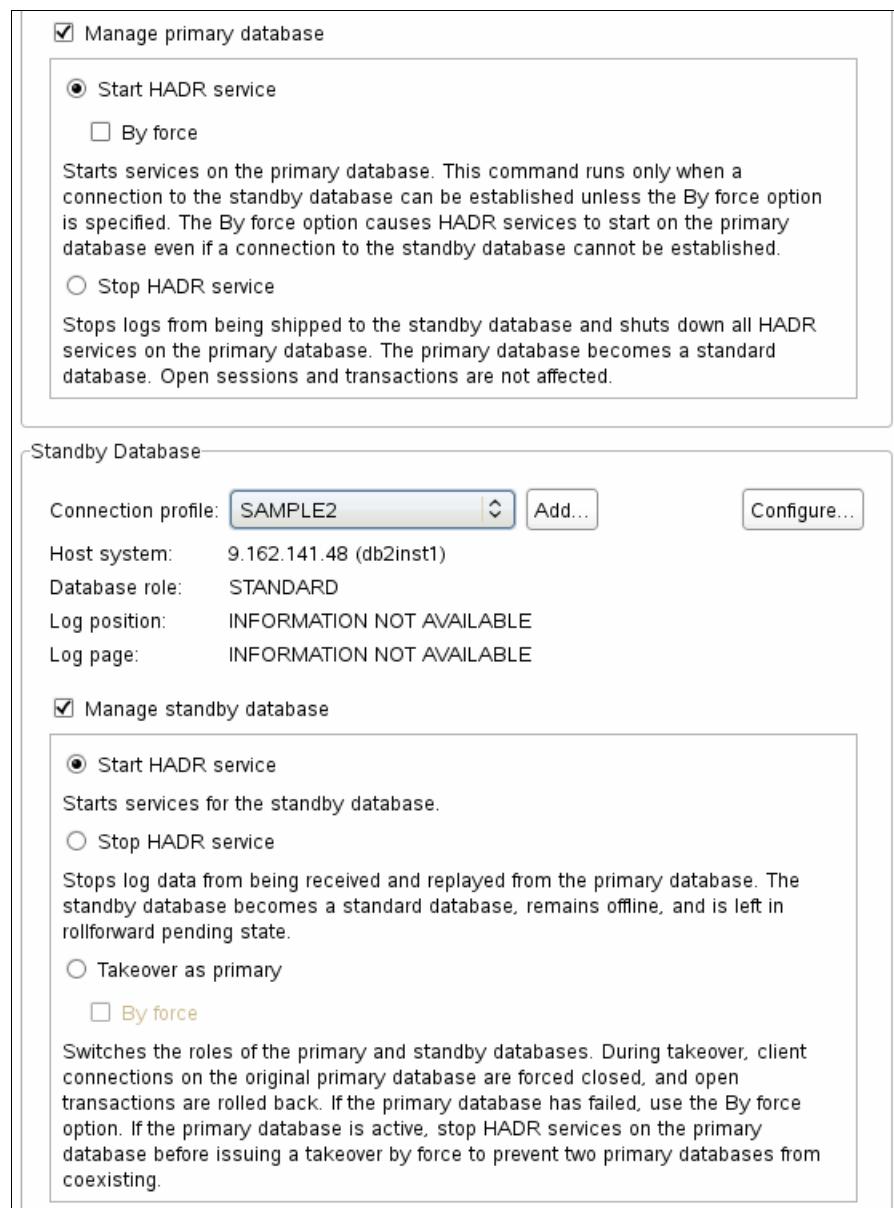


Figure 6-15 Start HADR from the Manage HADR window

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 run **STOP HADR**. If you run **STOP HADR**, the database becomes a standard database and might require reinitialization to resume operations as an HADR database. Instead, run **DEACTIVATE DATABASE**.

If you want to shut down only the HADR operation, this is the preferred 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 IBM Data Studio. When you want to bring your database to a standard database, run **STOP HADR**.

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 *DB2 10.1 Command Reference*, SC27-3868-00.

Stopping HADR from IBM Data Studio

You can stop HADR from IBM Data Studio by completing the following steps:

1. From IBM Data Studio, expand the **All Databases** tree in the Administration Explorer down to the wanted database object. Right-click the database that you plan to stop HADR on and select **Manage Database** → **Manage HADR** (Figure 6-14 on page 203).

- From the Manage High Availability Disaster Recovery (HADR) window, check the **Manage primary database** check box and select the **Stop HADR service** option (Figure 6-16). You can also stop just the standby, or both the primary and the standby, by repeating the process for the standby section of the HADR window (Figure 6-17 on page 207). Click **Run** to stop HADR.

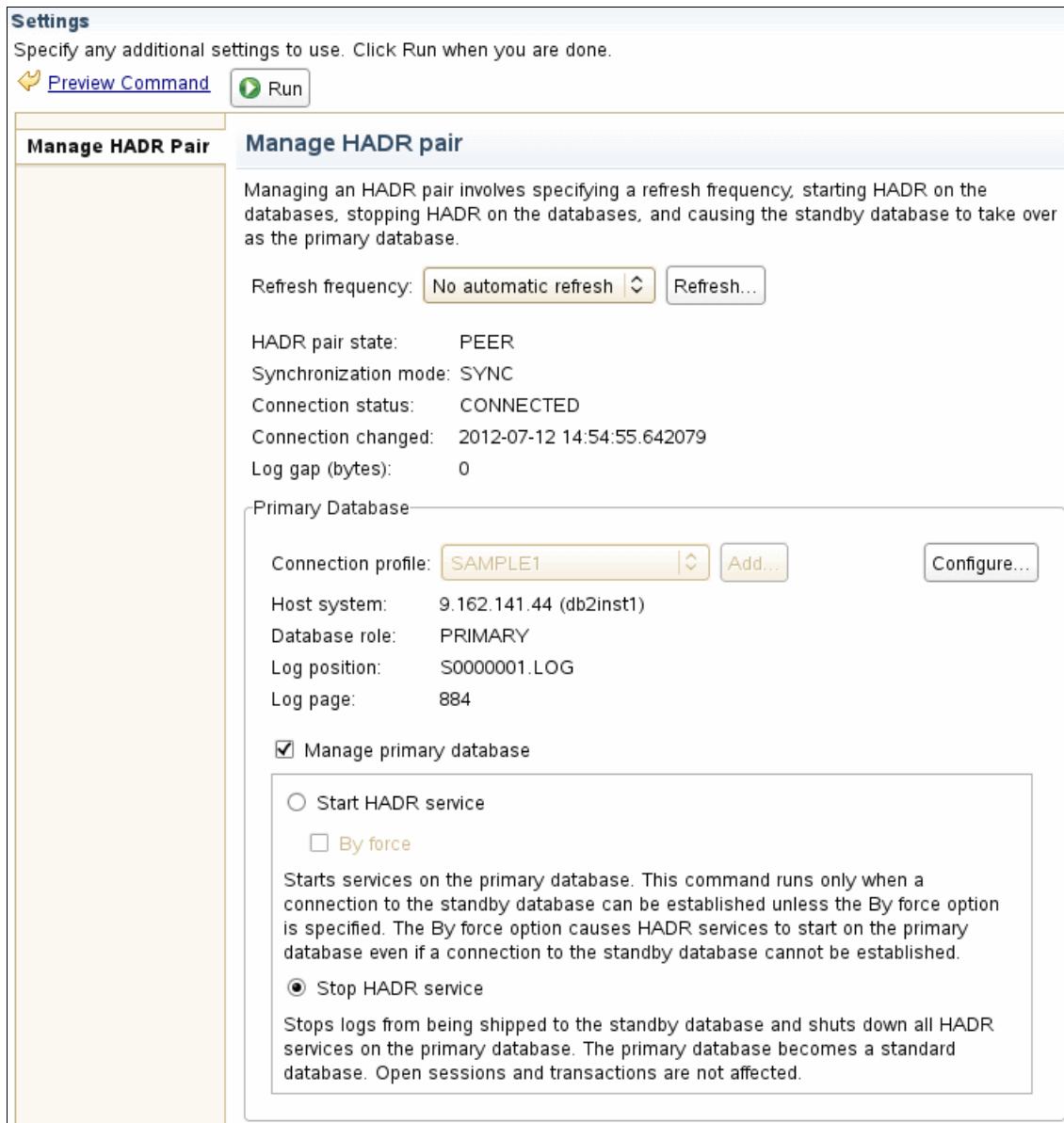


Figure 6-16 Stop HADR from the Manage HADR window

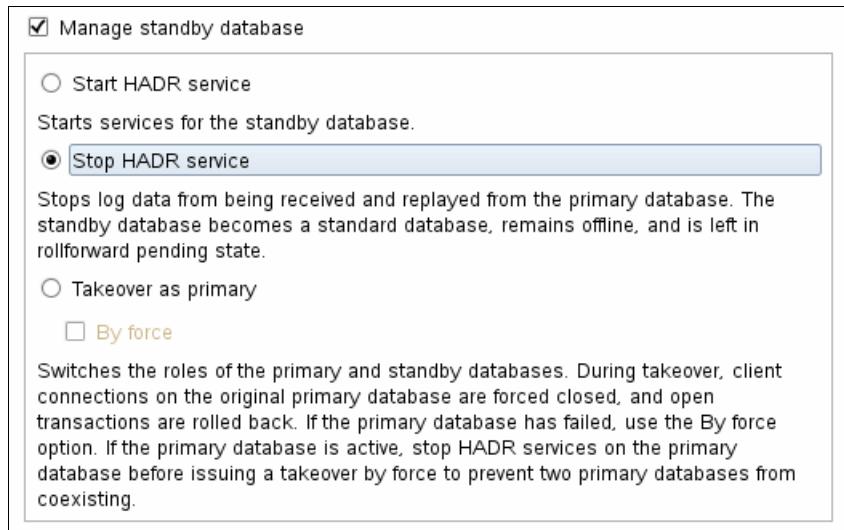


Figure 6-17 Manage standby database section of the HADR window

6.3.2 Planned takeover

Planned takeover is also referred to as switching roles or role exchange. You run **TAKEOVER** 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 reroute your clients either manually or by using ACR after you run **TAKEOVER**.

Switching roles is done only 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 to run **TAKEOVER HADR**. For example, run the **takeover** command on the standby database by running the following command:

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

The details for the **TAKEOVER HADR** command are found in *DB2 10.1 Command Reference*, SC27-3868-00.

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

1. The standby tells the primary that it is taking over.
2. The primary forces off all client connections and refuses new connections.
3. The primary rolls back any open transactions and ships the remaining log, up to the end of the log, to standby.

4. The standby replays the received log, up to the end of the log.
5. The primary becomes the new standby.
6. The standby becomes the new primary.

Using IBM Data Studio

In our example, we start out with the SAMPLE database on NODE2 as the primary database. To perform a takeover in IBM Data Studio, complete the following steps:

1. From IBM Data Studio, expand the **All Databases** tree in the Administration Explorer down to the wanted database object. Right-click the database that you plan to take over HADR and select **Manage Database** → **Manage HADR** as shown in Figure 6-14 on page 203.
2. In the HADR window, check the **Manage standby database** check box and select the **Takeover as primary** radio button (Figure 6-18).

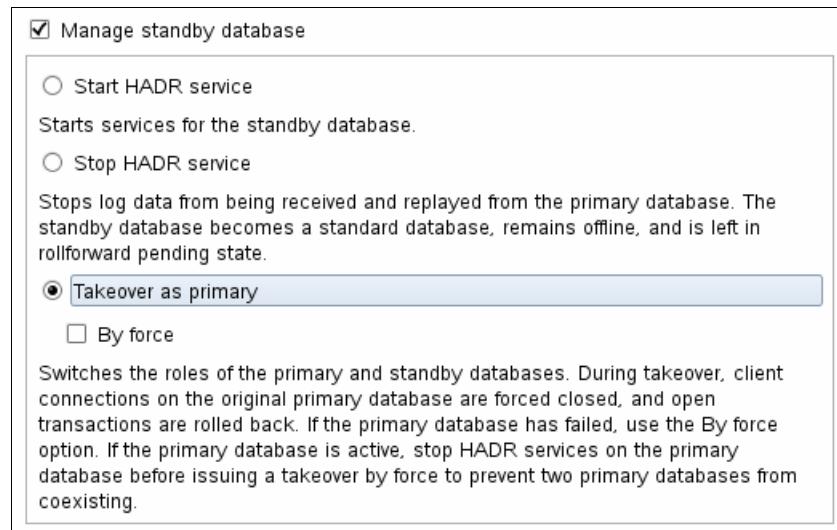
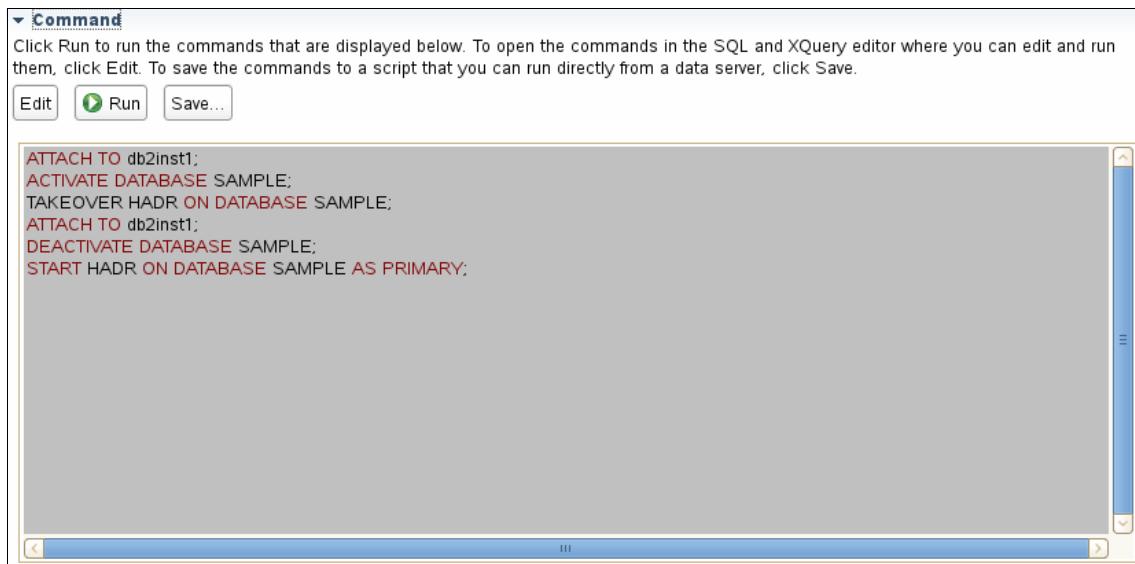


Figure 6-18 Manage HADR Takeover

3. Clicking the **Preview Command** link shows the commands DB2 that runs for the takeover operation (Figure 6-19).



The screenshot shows a command window titled 'Command'. It contains a text area with the following SQL script:

```
ATTACH TO db2inst1;
ACTIVATE DATABASE SAMPLE;
TAKEOVER HADR ON DATABASE SAMPLE;
ATTACH TO db2inst1;
DEACTIVATE DATABASE SAMPLE;
START HADR ON DATABASE SAMPLE AS PRIMARY;
```

Below the text area are three buttons: 'Edit', 'Run' (which is highlighted), and 'Save...'. The window has standard OS X-style scroll bars on the right and bottom.

Figure 6-19 Command windows shows the Takeover HADR script

4. Click **Run** to cause the standby to take over as primary.

The primary and the standby roles are swapped. Run the same command again to swap the roles back. This action can be done as often as necessary, assuming that the databases are in the Peer state.

6.3.3 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 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** suboption is specified, there is no committed transaction loss if the command succeeds and the primary database is stopped before the end of the peer window period.

The procedure for failover includes the following steps:

1. Ensure that the primary is down to minimize the chances of data loss. If a takeover by force is run and the primary is not down, this action can result in both the databases becoming primary databases. This situation is referred to as *split-brain*.
2. Run **TAKEOVER HADR** with the **BY FORCE** and **PEER WINDOW ONLY** options on the standby database. Here is an example of this command for the SAMPLE database:

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

After you run **TAKEOVER HADR** with the **BY FORCE** option from the standby, the following steps are carried out in the background:

1. The standby sends a notice that tells the primary to shut down.
2. The standby does *not* wait for any acknowledgement from the primary to confirm that it received the takeover notification or that it shut down.
3. The standby stops receiving logs from the primary, finishes replaying the logs that it 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 loses only transactions that are committed by the primary if the primary and standby databases fail at the same time. This scenario is highly unlikely.
 - With syncmode set to ASYNC or SUPERASYNC, the standby database can lose transactions that are 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 with 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 four syncmodes (SYNC, NEARSYNC, ASYNC, and SUPERASYNC), transactions that are not received and processed by the standby database are lost.

When you run **TAKEOVER BY FORCE PEER WINDOW ONLY** and it succeeds, then there is not any transaction information on the primary database that is not copied to the standby database. This situation ensures a greater degree of data consistency.

For more information about ways to prevent data loss in a forced takeover, see Chapter 7, “Developing a backup and recovery strategy”, in *DB2 10.1 Data Recovery and High Availability Guide and Reference*, SC27-3870-00.

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, complete the following steps:

1. Recover the failed primary, and bring it back online.
2. Restart the failed primary as the new standby by running **START HADR**.
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 with the new standby fails. You must restore a backup of the current primary to your failed primary to start it as the new standby. For more information about reintegration, see 6.4.5, “Re-establishing HADR after failure” on page 217.

Example of takeover by force in IBM Data Studio

To perform a takeover by force in IBM Data Studio, complete the following steps:

1. From the IBM Data Studio, expand the **All Databases** tree in the Administration Explorer down to the wanted database object. Right-click the database that you plan to execute a takeover HADR on and select **Manage Database** → **Manage HADR**, as shown in Figure 6-14 on page 203.

2. In the HADR window, check the **Manage standby database** check box and select the **Takeover as primary** radio button (Figure 6-20). Check the **By force** check box.

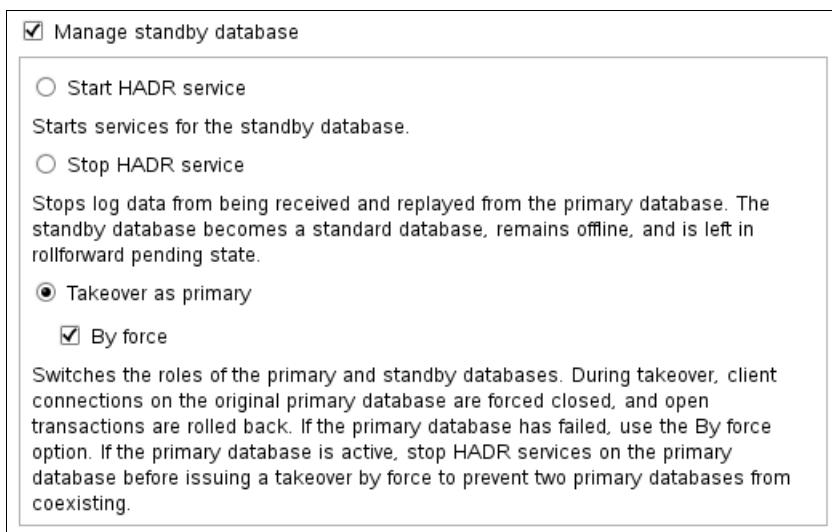


Figure 6-20 Takeover HADR by force

3. Click **Run** to force the standby to take over as the primary.

6.4 Troubleshooting

Here we examine some of the possible issues you might encounter while you set up and run HADR, and while you attempt to resume the HADR operation after a server failure, or another unanticipated interruption to DB2 HADR communication. In most cases, the solutions and workarounds are straightforward, but are not often immediately obvious from the error messages you might be getting at the time.

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

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.qb.server.doc/doc/r0025127.html>

6.4.1 During setup

Here is a list of issues and their possible fixes that might be encountered during setup:

- ▶ Symptom

On Windows, various error messages are claiming that the user ID does not have the authority to run a command, even though the user ID is both the DB2 instance owner 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 when DB2 was installed. In an Active Directory configured server, this user is most likely 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 owner 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 standby server when you attempt to discover DB2 instances and databases.

- Fix

Manually catalog the administrative TCP/IP node, the instance TCP/IP node, and the remote database on the command line, as shown in Example 6-3.

Example 6-3 Catalog the TCP/IP node and database

```
db2 catalog tcpip node db2_NODE2 remote x.x.x.x server 60004  
remote_instance db2inst1 system NODE2  
db2 catalog database sample as samp1_NODE2 at node db2_NODE2
```

6.4.2 After setup or during normal execution

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

- ▶ Symptom

A SQL30081N communication error is detected when you attempt to connect to the database on the remote server, even though there was no issue previously, and the relevant DB2 instance is definitely up and running on the remote server.

- Fix:

This issue is a common one 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, incorrect host name, or DNS issues. The first thing to do is see Chapter 26, “” in *DB2 10.1 Message Reference Volume 2*, SC27-3880-00 and perform any actions that it provides for any protocol-specific error codes. Then, you might stop using 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

After you change **db cfg** parameters on both servers and perform deactivation and reactivation on the standby, HADR still does not reconnect, so roles cannot be switched for the second database deactivation/reactivation.

- Fix

If you are changing certain parameters, such as any of the HADR_prefixes db cfg parameters, the currently active values must match on both sides for HADR connectivity to work. You require downtime on the primary after you change the db cfg parameter on both the servers to establish the new value as the current value. Plan for an outage window to accomplish this task, or 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.

6.4.3 After an HADR disconnects or server failure occurs

Here is an issue and possible fix that is 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 is recycled without any warning sent to 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 mismatched or lost log records on the standby, such that it cannot re-integrate successfully with the primary. You can discover if this is the case by 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 situation are clear. Subsequent corrective actions are 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 task can be found in 6.4.5, “Re-establishing HADR after failure” on page 217.

6.4.4 Considerations while running HADR

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

- ▶ If you schedule backup and maintenance 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 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 is 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 that control HADR databases. To explain this situation using an example of server A and server B, the situation is that DB2 on a broken primary server A does not automatically restart when server A is later fixed and started, causing a split-brain scenario.

Fortunately, DB2 HADR is able to protect itself from split-brain scenarios where an HADR primary database is deactivated while it is 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 administrative perspective to start the HADR standby database first and 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 deactivated (for example, the failure is with the network rather than with the server), and after a **db2 takeover hadr on database ... by force** command is run by the standby server B, the broken primary server A unexpectedly comes back online.

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

- Where the broken HADR PRIMARY on server A has a **db2 stop hadr on database** command run on it, is now in STANDARD mode, and clients can once again connect to it instead of the correct HADR primary database that is 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 connect to only a single server that has the role of the active node.

For more information about clustered environments, see Chapter 2, “DB2 with IBM Tivoli System Automation for Multiplatforms” on page 19, Chapter 3, “DB2 and PowerHA SystemMirror” on page 71, and Chapter 4, “DB2 with Microsoft Windows Failover Cluster” on page 101.

6.4.5 Re-establishing HADR after failure

Also referred to as *reintegration* of HADR, this process is similar to the initial setup of HADR, but without configuring the **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 a rapid and low complexity solution.

For our example, we use the SAMPLE HADR database 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 works only if the primary and the standby were in the Peer state at the time of server A’s failure. In this scenario, the DB2 instance is eventually restarted on server A after you correct whatever failure occurred. DB2 HADR protects itself from a split-brain scenario 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 now.

The next thing to do is run **db2 start hadr on database sample as standby** on Server A. The SAMPLE database cannot have a **db2 stop hadr on database sample** command that is run on it beforehand, as the database 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 a 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 a 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 the Peer state, consider this situation before you run a **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 run a takeover by force command on Server B's STANDBY database. The Peer state is the only guarantee you have that no data is lost from the committed transactions.

The second way works regardless of whether the HADR pair was in a 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 is dropped before the restore.

If you must drop your standby database before you run the restore from primary backup, you must 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 before the restore. However, DB2 does not let you restore over the top of a database that is in an HADR standby or primary state. In some cases, DB2 does not let you stop HADR and put that database into a standard mode; this situation can happen if HADR was not stopped on a database before the DB2 instance is migrated from Version 9.7 to Version 10.1, for example.

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

If you are concerned about scheduling a downtime window for re-establishing HADR, be 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 stated in Chapter 3, “Command line processor plus (CLPPlus)”, in *DB2 10.1 Command Reference*, SC27-3868:

“During an online backup, DB2 obtains IN (Intent None) locks on all tables that exist in SMS table spaces as they are processed. S (share locks) are no longer held on LOB data in SMS table spaces during online backup.”

Command-line steps

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

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

Back up your primary database. You might want to compress the backup to save time when you transfer it to the remote (standby) server, or use a Tivoli Storage Manager target that the remote server is authorized to read from. In our example on Linux, we make a compressed backup of the SAMPLE database on the local server (NODE1) to local disk, then use **scp** to transfer the backup file to the remote server (NODE2):

```
db2 backup database sample online to /usr/db2/backup compress  
Backup successful. The timestamp for this backup image is :  
20120716153706
```

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

For our example, we must discover the name of the backup file; the destination of the backup step was /usr/db2/backup. The backup file name on Windows matches Linux/UNIX, a long file name in a single subdirectory.

Example 6-4, shows the results of running **ls -l** on the /usr/db2/backup in the Linux server.

Example 6-4 List database backup files

```
b2inst1@node1:/usr/db2/backup> ls -l  
-rw-r----- 1 db2inst1 db2grp1 12079104 2012-07-16 15:37  
SAMPLE.0.db2inst1.NODE0000.CATN0000.20120716153706.001
```

The backup file name that we want contains a matching timestamp with the output of the example backup command. The matching file for us is:

SAMPLE.0.db2inst1.NODE0000.CATN0000.20120716153706.001

The timestamp (20120716153706 in our example), is also used in the restore command later, so note it.

Now, we actually run the transfer command. We use **scp** in our example, but you could use **sftp** or a similar command:

```
scp SAMPLE.0.db2inst1.NODE0000.CATN0000.20120716153706.001  
db2inst1@node2:/usr/db2/backup
```

3. We log on to the remote server (NODE2 in our example) and run the restore database command. HADR is not able to re-establish a connection if the standby database is rolled forward after the restore.

```
db2 restore db sample from /usr/db2/backup taken at 20120716153706
```

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

4. The final step involves starting HADR on both the primary and standby, and checking that HADR is connecting successfully. You can wait until the standby database reaches the Peer state, or leave it in the catchup-pending state, where it eventually catches up and reaches the Peer state.

Complete the following steps:

- a. In our example, the primary database never left the HADR primary state, so we need only to start HADR on the standby by running the following command:

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

If HADR is not started on the primary server database, we could achieve this start by logging on there and running the following command:

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

Restarting HADR: If HADR is stopped on both servers for a database, restart the standby first, then the primary, or you end up with this error message:

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

- b. To check that HADR is connected and working, run either **db2diag.log** or **db2pd**:

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

The output of these commands for our example is shown in Example 6-5.

Example 6-5 Checking HADR using the db2pd output

```
Database Member 0 -- Database SAMPLE -- Active -- Up 6 days  
00:07:31
```

```
HADR_ROLE = STANDBY
```

```

HADR_STATE = PEER
HADR_SYNCMODE = SYNC

HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 11/07/2012 12:49:59.216304
(1342007399)
HADR_TIMEOUT (seconds) = 120

PRIMARY_MEMBER_HOST = NODE2
PRIMARY_INSTANCE = db2inst1

STANDBY_MEMBER_HOST = NODE1
STANDBY_INSTANCE = db2inst1

PRIMARY_LOG_FILE,PAGE,POS = S0000004.LOG, 900, 60732812
STANDBY_LOG_FILE,PAGE,POS = S0000004.LOG, 900, 60732812

```

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

db2 takeover hadr on db sample

In our example environment, a subsequent running of **db2pd** shows the results in Example 6-6.

Example 6-6 db2pd output - HADR is in the Peer state

```

Database Member 0 -- Database SAMPLE -- Active -- Up 6 days
00:07:31

HADR_ROLE = PRIMARY
HADR_STATE = PEER
HADR_SYNCMODE = SYNC

HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 11/07/2012 12:49:59.216304
(1342007399)
HADR_TIMEOUT (seconds) = 120

PRIMARY_MEMBER_HOST = NODE1
PRIMARY_INSTANCE = db2inst1

STANDBY_MEMBER_HOST = NODE2
STANDBY_INSTANCE = db2inst1

```

```
PRIMARY_LOG_FILE,PAGE,POS = S0000004.LOG, 903, 60744782
STANDBY_LOG_FILE,PAGE,POS = S0000004.LOG, 903, 60744782
```

- d. If you are managing HADR from IBM Data Studio, you would catalog the remote node and database aliases, so alternatively, you can run the takeover command remotely by specifying the remote database alias and the explicit DB2 instance user ID and password. For example, if we now want to switch the roles again so that SAMPLE database on the NODE1 server becomes the primary, we can run the following command from the NODE2 server:

```
db2 takeover hadr on db samp1_NODE1 user db2inst1
```

- e. If you cataloged only 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 on the standby by using the command line processor (CLP).

In summary, re-establishing (reintegrating) HADR should be as simple as restoring a backup from the primary over the top of the broken standby, running the appropriate start HADR commands, and downtime is not required.



HADR with clustering software

Clustering software, such as PowerHA SystemMirror (PowerHA) and IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP), are 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 describe how to configure HADR with PowerHA or Tivoli SA MP to enable automating HADR takeover.

This chapter covers the following topics:

- ▶ Overview: Why clustering software is needed
- ▶ db2haicu
- ▶ DB2 HADR with Tivoli SA MP configuration for automatic failover on an AIX system
- ▶ DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system
- ▶ Automating HADR takeover with PowerHA

7.1 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 a down network. As illustrated in Figure 7-1, 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 run the appropriate takeover commands if there is a primary database server failure. This situation is where clustering software is required to automate HADR takeover.

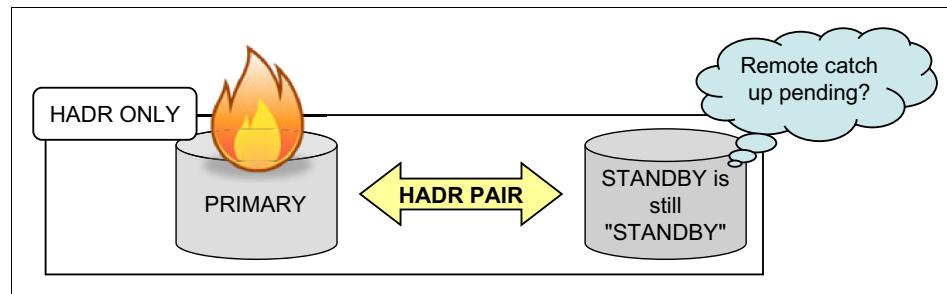


Figure 7-1 Takeover HADR does not happen automatically

7.1.1 What is clustering software

Clustering software automates the 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 where 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 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 SA MP**

Tivoli System Automation is a solution that is designed to provide a multitiered architecture of clustering in a heterogeneous environment. This architecture can be realized either through the Tivoli SA MP base component on various platforms, or through existing base-level cluster solutions from other vendors, which are coupled with the Tivoli SA MP end-to-end component, which interfaces with all the base-level clustering software (with adapters for non Tivoli SA MP products). This setup enables central control of all clusters, and facilitates the grouping of interdependent base-level clusters into applications. Tivoli SA MP 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 2, “DB2 with IBM Tivoli System Automation for Multiplatforms” on page 19.

- ▶ **IBM PowerHA SystemMirror for AIX**

PowerHA offers robust high availability and disaster recovery for IBM customers with mission-critical applications. PowerHA provides base services for cluster node membership, system management, configuration integrity and control, and failover and recovery for applications. PowerHA clusters with both non-concurrent and concurrent access can be an IBM System p®, or a logical partition (LPAR) of an applicable System p.

For detailed information about PowerHA, see the following website:

<http://www-03.ibm.com/systems/power/software/availability/aix/index.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: You can use it to connect up to eight nodes in a cluster; more technology is added to make geographically remote clusters possible.

For information about MSCS, see the white paper *Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server*, found at:

<http://ibm.com/software/data/pubs/papers/>

- ▶ LifeKeeper for Linux/Windows

LifeKeeper for Linux/Windows is a clustering software that is 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 high availability for Linux/Windows applications.

For more information about this product, go to:

<http://www.steeleye.com/products/>

DB2 UDB and SteelEye LifeKeeper for Linux - A High Availability Database, introduces a sample configuration of integrating DB2 into a LifeKeeper cluster environment. This article is available at:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/lk.pdf>

7.1.2 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 start HADR takeover on the standby database with the resource failover. Figure 7-2 shows a typical failover flow in the HADR cluster that is automated by the clustering software.

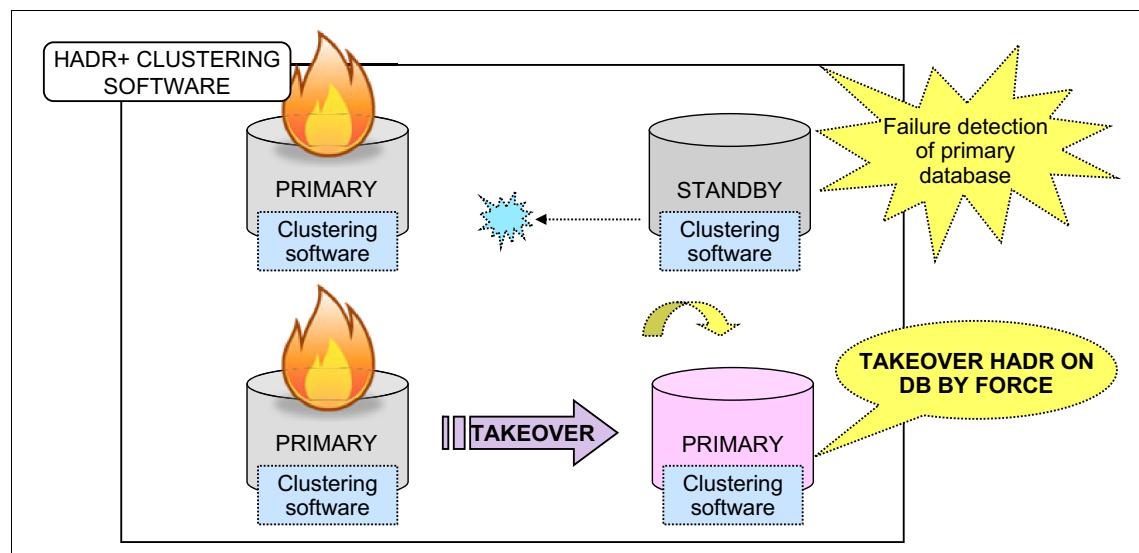


Figure 7-2 Why clustering software is needed to automate HADR takeover

In the cluster environment, the typical failover flow is as follows:

1. Under normal operations, users connect to the primary server and run 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, therefore replicating the primary database. The standby database is in rollforward pending and no user can access this database.
2. If there is an outage on the primary server, clustering software detects the fault, starts to fail over the defined resources, such as an IP address, and run 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 scripts run 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. Although 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 access is achieved by the DB2 automatic client reroute (ACR) feature. Whenever the client fails to connect to the original server, the DB2 client attempt to reroute the connection to the alternate server. If the reroute succeeds, the application can continue.
5. When the failed server comes back online, it can then be started as the standby server, reversing the roles to the ones that existed before the outage.

7.1.3 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 are on separate storage devices.

- ▶ IP address

IP address takeover is optional. When you use ACR, 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 that communicate with the database server and do not support ACR, 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 you detect an outage on the primary database.

7.2 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 run time at **db2haicu** prompts.

7.2.1 Prerequisites

Before you run **db2haicu**, there is a set of tasks that must be performed by a user with root authority and that database manager instance owner.

A user with root authority must initialize **db2haicu** on all machines that are added to the cluster domain by running the following command:

```
preprnode
```

This command prepares security on the node on which the command is run so it can be defined in a peer domain. It allows 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 needs only 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 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 by running **db2start**.

There are some tasks specific to 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 with a value of at least 120 seconds.
- ▶ Disable the DB2 fault monitor.

For a partitioned database environment, before you configure it for high availability, configure the DB2_NUM_FAILOVER_NODES registry variable on all machines that are added to the cluster domain. This variable specifies the number of additional database partitions that might need to be started on a machine in the event of failover.

When all of these tasks are done, a database manager instance owner can use **db2haicu** to perform cluster configuration and administration operations.

7.2.2 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 Tivoli SA MP Cluster Manager. During this configuration process, all necessary resources, dependencies, and equivalencies are automatically defined to Tivoli 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 that is called DPF.xml that has all cluster definitions, you can run the following command to create a cluster:

```
db2haicu -f DPF.xml
```

► **-disable**

Use this option to disable the HA configuration for an instance. After you run the 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 that are 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 that identify 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 that are 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 that are associated with the current database manager instance, 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 run **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, run **db2haicu** 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 that are provided in the file. This option is 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`, run the following command:

```
db2haicu -f db2haicu-sample.xml
```

There is a set of sample XML input files that are in the `samples` subdirectory of the `sql1ib` 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 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 that is created and configured for the instance, **db2haicu** begins the process of creating and configuring a cluster domain. While creating the domain, **db2haicu** prompts you 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 that is 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 the following tasks:

- 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 the failover policy for this instance.
- Create a quorum device for the domain.
- Destroy the domain.
- Exit.

7.2.3 Considerations

Consider the following list of practices for configuring your cluster and your database manager instances when you use **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 synchronization is critical to ensure a smooth failover.

For an automatic instance failover, it is a preferred practice (but it is not mandatory) that the time and dates on cluster nodes be synchronized.

For information about how to configure NTP for your system, see your operating system documentation.

- ▶ For an automatic instance failover cluster configuration, when you add 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
<code>/dev/sdd</code>	<code>/shared_home</code>	<code>ext3</code>		<code>noauto</code>

- ▶ Any maintenance work can be performed by disabling the HA configuration by running **db2haicu -disable** without worrying about cluster manager intervention. After you run this command, the system does not respond to any failures, and all resource groups for the instance are locked.

7.2.4 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 complete the following steps:

- ▶ Remove the resource groups of the partially created cluster domain by running **db2haicu** with the **-delete** parameter.
- ▶ Re-create the cluster domain by calling the **db2haicu** utility again.

7.3 DB2 HADR with Tivoli SA MP configuration for automatic failover on an AIX system

This section describes how to configure automatic DB2 HADR takeover with Tivoli SA MP using the **db2haicu** command in an AIX environment.

In an HADR with Tivoli SA MP environment, Tivoli SA MP 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 Tivoli SA MP cluster, HADR is placed under Tivoli SA MP control. For more information about Tivoli SA MP, see Chapter 2, “DB2 with IBM Tivoli System Automation for Multiplatforms” on page 19.

7.3.1 Architecture

Before you set up an HADR with Tivoli SA MP 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) provides regular 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 that is issued by administrators.
 - Confirm the software requirements on each node.
- ▶ Network:
 - A network interface that provides client access on each node. This interface is one of the Tivoli SA MP controlled resources.
 - Have one network card for a Tivoli SA MP tiebreaker. Have a separate network for HADR to avoid interference by the HADR log transfer.

- ▶ Tivoli SA MP configuration:
Define a cluster domain name that includes the resources of a virtual IP (service IP), a DB2 instance, and a 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. To define the HADR configuration, see Chapter 10, “Automatic client reroute” on page 377.
- ▶ Scripts:
DB2 10.1 automatically installs the Tivoli SA MP scripts that are used for start, stop, and monitoring DB2 resources.

Lab environment

Figure 7-3 shows the configuration of HADR with Tivoli SA MP in our lab environment.

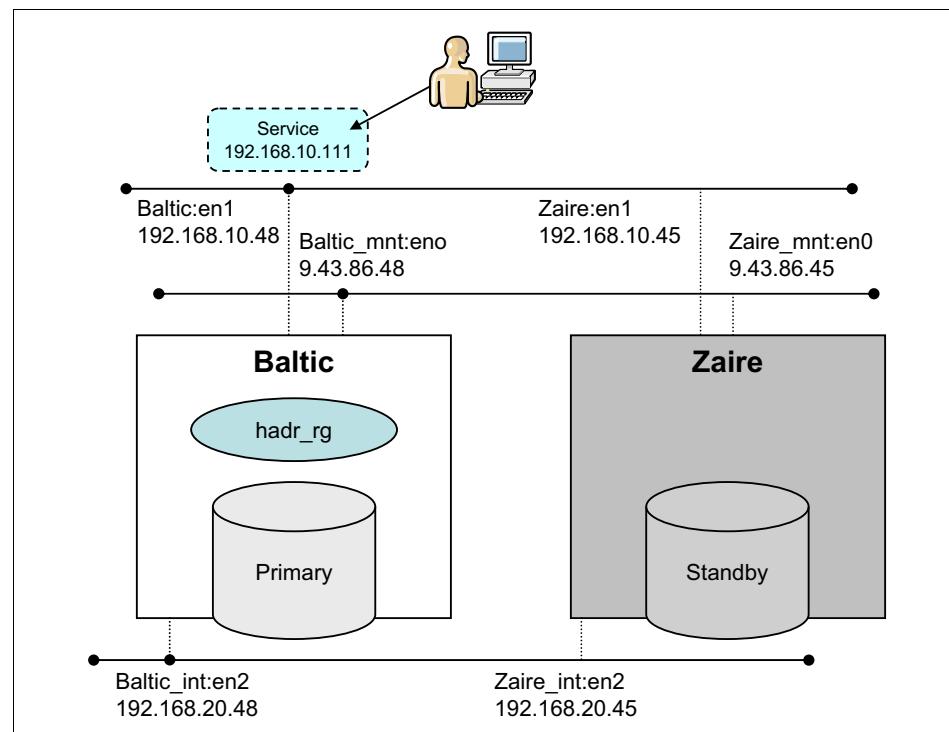


Figure 7-3 HADR with Tivoli SA MP lab environment

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 a *service node* and has the HADR primary database.
 - Zaire is the *standby* node on which the HADR standby database is.
 - The software configuration that is used to set up the lab environment includes:
 - AIX 7.1 Technology Level 1
 - DB2 10.1 Advanced Enterprise Server Edition
 - Tivoli SA MP Base Component V3.2
- ▶ Network configurations:
 - One Ethernet network interface (en1) on each node is provided for client access, under the control of Tivoli SA MP. 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 Tivoli SA MP tiebreaker. For a detailed description about the Tivoli SA MP tiebreaker, see 2.1.2, “Terminology of Tivoli SA MP” on page 22.

Shared disks: For HADR in a Tivoli SA MP cluster, shared disks are not necessary because the primary and standby databases are independent databases, which can be in separate storage devices.

- ▶ Tivoli SA MP configuration:
 - A Cluster Domain named hadr_domain is configured, which includes the virtual IP (service IP), DB2 instance, and HADR database.
 - In our lab, we configured a virtual IP in the Tivoli SA MP 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.
 - A configured dedicated network for HADR communication.

7.3.2 Configuration

In this section, we provide the steps to configure an automated HADR takeover environment with Tivoli SA MP. HADR setup must be done before you run **db2haicu**.

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 the correct entries are configured in the /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 the HADR parameters correctly 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 by running **db2pd**.

For a step-by-step configuration of HADR, see Chapter 6, “HADR setup” on page 173.

Figure 7-4 shows the entries of the services and hosts of the HADR configuration in our lab environment.

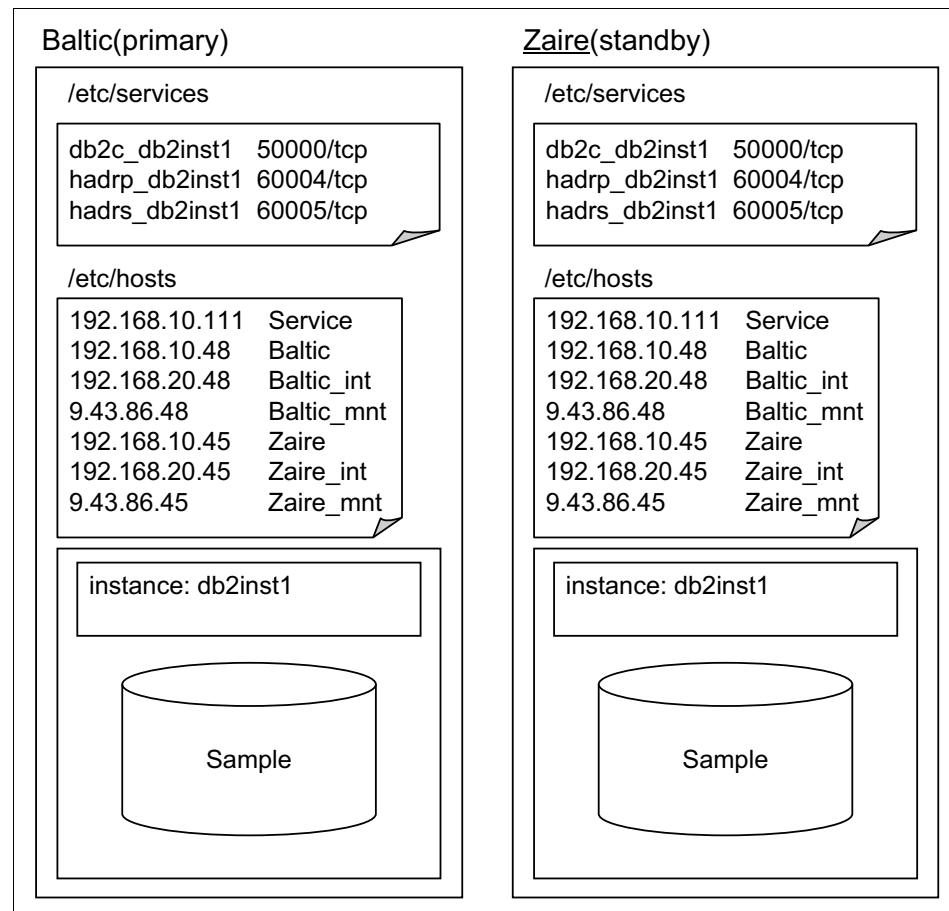


Figure 7-4 Configuration of HADR

Checking the Internet Protocol network connection

Before you configure Tivoli SA MP, check that the network is configured correctly. To check network configurations for the cluster, complete the following steps.

In our examples, we use the notations (P) and (S) proceeding a command to designate on which node that command is to be issued to properly set up the topology that is shown in Figure 7-4. The order of the notation also designates the sequence of the command execution on the nodes.

The notations are:

- ▶ (P): Primary database node (Baltic)
 - ▶ (S): Standby database node (Zaire)
 - ▶ (P)(S): Command that is issued on the primary node first, then the secondary node
1. Check whether the IP addresses are configured on network interfaces by running the following command:

```
netstat -in | grep -v link
```

Example 7-1 shows the configuration of the IP addresses on Baltic.

Example 7-1 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 7-2 shows the content of our hosts files.

Example 7-2 hosts file content

```
(P)(S)#vi /etc/hosts
192.168.10.111  Service    ## Virtual IP address
192.168.10.48   Baltic      ## SA MP 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       ## SAMP 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 by running **host**. 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 Tivoli SA MP by using db2haicu

The main steps for this setup are as follows:

1. Complete the Initial configuration.
2. Configure the cluster domain and resource by using **db2haicu**.

Next, we provide the detailed steps for this setup.

Initial configuration

On both nodes, run **preprnode** as root to prepare the local node to join the domain:

```
(P)(S) # preprnode Baltic Zaire
```

Configuring the cluster domain and resource by using db2haicu

In this section, we demonstrate the cluster configuration by using the **db2haicu XML file setup mode**. For information about configuring with **db2haicu** interactive setup mode, see 7.4, “DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system” on page 266.

Ensure that all the nodes to be configured in the SAMP cluster domain can see each other in the network and can communicate with each other.

Run **db2haicu** 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 make a DB2 HADR instance cooperate with Tivoli SA MP. DB2 provides sample XML files that are in the sqllib/samples/ha/xml directory. These sample XML files can also be found at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0052514.html>

Example 7-3 illustrates a sample db2haicu XML file.

Example 7-3 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" />
values for local and remote hosts remain the same or they must be
swapped inside xml file for primary node??????
<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 information 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>* subelement of the ClusterDomain element specifies the quorum device for the cluster domain.
 - The *<PhysicalNetwork>* subelement of the ClusterDomain element includes all network information. This information includes the name of the network and the network interface cards that are contained in it. We define our single public network and private network using this element.
 - The *<ClusterNode>* subelement 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 that is 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 that is 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 the 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 information includes the current DB2 instance name and the DB2 partition number.
- The *<HADRDBS>* element covers the HADR database information. This information includes the primary node name, standby node name, primary instance name, standby instance name, and the virtual IP address that is associated with the database.

For a more detailed description of the XML files, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/r0053130.html>

2. Run **db2haicu**.

db2haicu must be run first on the standby instance and then on the primary instance for the configuration to complete, as follows:

`db2haicu -f XMLfilepath`

Example 7-4 shows the sample output of **db2haicu** with XML on the standby node.

Example 7-4 Sample output of db2haicu with XML on the standby node

(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 apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu 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 7-5 shows the sample output of **db2haicu** with XML on the primary node.

Example 7-5 Sample output of db2haicu with XML on the primary node

(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 apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu 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 ...
```

Primary node messages: The messages regarding the networks (bold text) encountered on the primary node can be safely ignored. These messages appear because we 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. Run **1ssam**, **1srpdomain**, and **1srpnode** to see the resources that are created during this process.

Example 7-6 shows the domain cluster resource status that is listed by running **1ssam**.

Example 7-6 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
```

Example 7-7 shows the cluster domain status that is listed by running **1srpdomain**.

Example 7-7 List cluster domain status

```
(P) # 1srpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
hadr_domain Online 3.1.2.2 No           12347 12348
```

Example 7-8 shows the nodes in the cluster that are listed by running **1srpnode**.

Example 7-8 List of nodes in the cluster

```
(P) # 1srpnode
Name  OpState RSCTVersion
baltic Online 3.1.2.2
zaire Online 3.1.2.2
```

Resource groups topology

After the `db2haicu` tool is run successfully on both the standby and primary instances, the setup is complete. The relationships among the resources are shown in Figure 7-5.

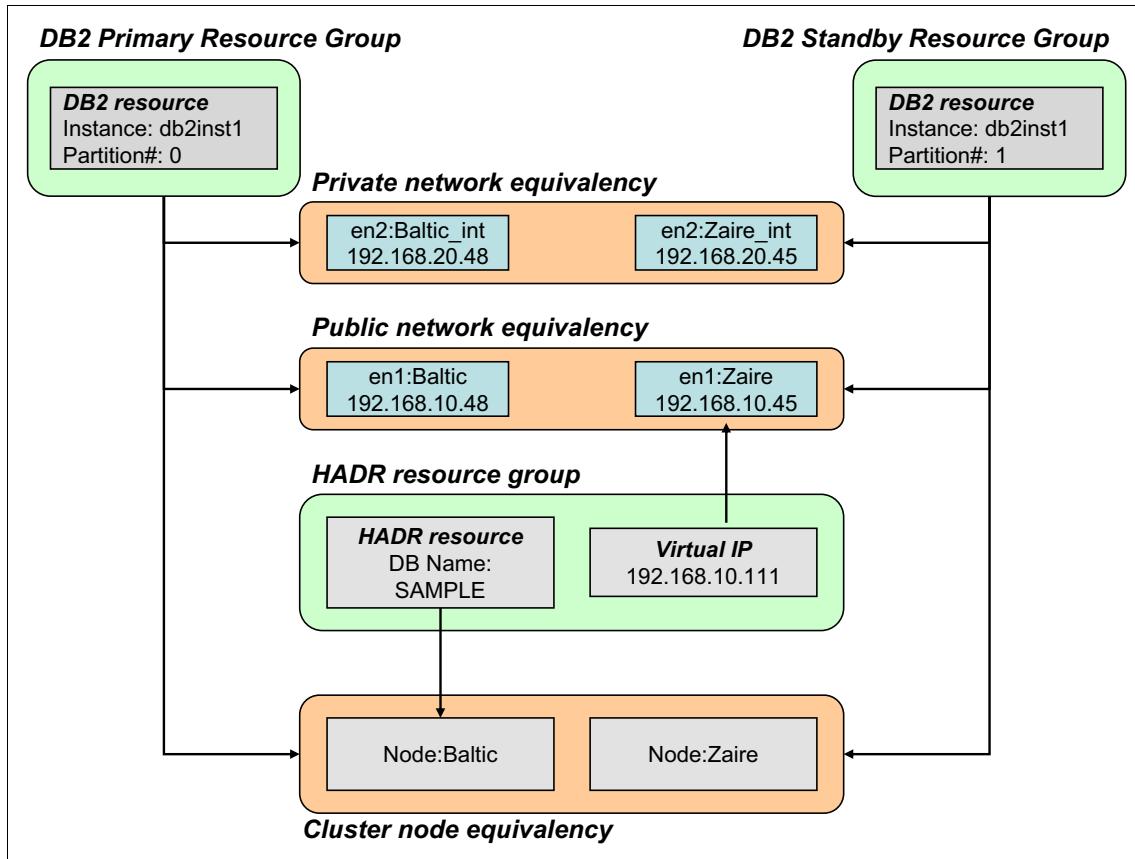


Figure 7-5 Resource groups that are created for a multiple network HADR topology

The following list shows how the resources illustrated in this figure correspond to the resources listed in the `lssam` command that is shown in Example 7-6 on page 245:

- ▶ Primary DB2 instance resource group: `db2_db2inst1_Baltic_0-rg (lssam)`
 - Member resources:
 - Primary DB2 resource: `db2_db2inst1_Baltic_0-rs (lssam)`

- ▶ Standby DB2 instance resource group: db2_db2inst1_Zaire_0-rg (**1ssam**)
 - Member resources:
 - Standby DB2 resource: db2_db2inst1_Zaire_0-rs
- ▶ HADR database resource group: db2_db2inst1_db2inst1_SAMPLE-rg (**1ssam**)
 - Member resources:
 - HADR DB resource: db2_db2inst1_db2inst1_SAMPLE-rs (**1ssam**)
 - Virtual IP address resource: db2ip_192_168_10_111-rs (**1ssam**)

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.

Run **db2haicu** with the **-delete** option on an instance before it is made highly available. This action ensures that the setup is started from scratch and there are no residuals from the previous build. For example, when you run **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 ensure that any temporary resources created during the initial run are cleaned up.

The **db2haicu -delete** option leaves the DB2 instances and the HADR replication unaffected. 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 7-9 shows a sample output of the **db2haicu -delete** command.

Example 7-9 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 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 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 ...

7.3.3 Administration

From time to time, you must plan for a system outage for some maintenance activities, such as a software upgrade or recycling a DB2 instance to change non-dynamic database manager parameters. In this section, we demonstrate how to perform some DB2, OS, and SAMP operations manually for a planned outage or maintenance.

When you complete the DB2 HADR with SAMP setup, you can perform these operations as some simple tests to verify your configuration.

Manual DB2 instance operations

Here we describe how to stop and start DB2 on the primary and the standby nodes for a planned outage or maintenance.

Running db2stop and db2start on the standby

You can run **db2stop** and **db2start** on the standby node without impacting the activities that are taking place on 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 the *Pending Online* state. 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 7-10 illustrates the effect of the **db2stop force** command that is issued on the standby instance.

Example 7-10 Issam output after you stop DB2 on the standby instance

```
(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
          Control=StartInhibitedBecauseSuspended
            '- 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
                  Control=SuspendedPropagated
                    |- 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
                          Control=SuspendedPropagated
                            |- 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, start the standby DB2 instance and activate the database again by running the following command:

```
(S) $db2start;db2 activate db sample
```

Running the db2stop and db2start commands on the primary

Running **db2stop force** on the primary instance breaks the connection from the client and halts HADR:

```
(P) $db2stop force
```

Example 7-11 illustrates the effect of the **db2stop force** command that is issued on the primary instance. The takeover does not occur because that is a planned operation.

Example 7-11 Issam output after you stop DB2 on the primary instance

```
(P) (S) #lssam
Pending online IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg Request=Lock
Nominal=Online
  '- Offline IBM.Application:db2_db2inst1_Baltic_0-rs
    Control=StartInhibitedBecauseSuspended
      '- 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
    Control=StartInhibitedBecause Suspended
      |- 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
    Control=SuspendedPropagated
      |- 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, start the primary DB2 instance and activate the database again by running the following command:

```
(P) $db2start;db2 activate db sample
```

Running takeover hadr on the standby instance

There might be situations when a DBA wants to perform a manual takeover to switch database roles. The safest way to accomplish this task is to run **takeover hadr** without the **by force** option.

Log on to the standby node and run **takeover hadr on db dbname** to perform a manual takeover. For example:

```
(S) $db2 takeover hadr on db sample
```

After the takeover completes successfully, running the **lssam** commands shows the changes. The virtual IP (service IP) address is also moved to the new primary node as part of the takeover process (Example 7-12).

Example 7-12 lssam 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

Here we describe OS and Tivoli SA MP operations for a planned outage or maintenance.

Stopping the standby node

If you need to stop Tivoli SA MP on the standby node for maintenance, complete the following steps:

1. Run **db2stop force** on the standby instance:

```
(S) $db2stop force
```

2. Run **stoprpnnode hostname** command as the root user on the primary node. If you run **stoprpnnode** on the standby node, you cannot issue any Tivoli SA MP or RSCT operations on the standby node. So, run these commands on the primary node:

```
(P) #stoprpnnode Zaire
(P) #lsrpnode
Name      OpState RSCTVersion
Zaire    Offline  3.1.2.2
Baltic   Online   3.1.2.2
```

3. Run **1ssam** to observe the state of the resources (Example 7-13).

Example 7-13 1ssam output after stoprnode

```
(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
Failed offline IBM.ResourceGroup:db2_db2inst1_Zaire_0-rg
Request=Lock Nominal=Online
    '- Failed offline IBM.Application:db2_db2inst1_Zaire_0-rs
        Control=SuspendedPropagated
            '- 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
        Control=SuspendedPropagated
            |- 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
        Control=SuspendedPropagated
            |- Online
IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
    '- Failed offline
IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire Node=Offline
```

4. You can perform the maintenance operations on the standby node, such as rebooting the OS. After the maintenance work is complete, run **startrpnode hostname** for the standby node on the primary node. For example:

```
(P) #startrpnode Zaire
(P) #lsrpnode
Name  OpState RSCTVersion
Zaire  Online 3.1.2.2
Baltic Online 3.1.2.2
```

5. Run **db2start** on the standby instance:

```
(S) $db2start;db2 activate db sample
```

6. Run **1ssam** to observe the state of the resources. Confirm that the resource of the instance on Zaire is online.

Stopping the primary node

If you must stop Tivoli SA MP on the primary node for the planned outage or maintenance, complete the following steps:

1. Switch the roles. Have the standby (Zaire) take over the services so you can stop the primary node (Baltic) by running the following command:

```
(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. Run **db2stop force** on the new standby node (Baltic):

```
(S) $db2stop force
```

3. Run **stoprpnode** as the root user for the new standby node. You run this command on the *new* primary node (Zaire):

```
(P) #stoprpnode Baltic
```

```
(P) #lsrpnode
```

```
Name OpState RSCTVersion
```

```
Baltic Offline 3.1.2.2
```

```
Zaire Online 3.1.2.2
```

4. Run **1ssam** to observe the state of the resources.

5. Now you can perform the maintenance operations on the new standby node, such as rebooting the OS. After the maintenance work is complete, run **startrpnode** for the new standby node (Baltic) on the new primary node (Zaire):

```
(P) #startrpnode Baltic
```

```
(P) #lsrpnode
```

```
Name OpState RSCTVersion
```

```
Baltic Online 3.1.2.2
```

```
Zaire Online 3.1.2.2
```

6. Run **db2start** on the new standby instance:

```
(S) $db2start;db2 activate db sample
```

7. Run **1ssam** to observe the state of the resources and confirm that the resources of the instance on Baltic are online.

8. Have the old primary node (Baltic) take over the services again by running the following command:

```
(S) $db2 takeover hadr on db sample
```

After the takeover completes successfully, check that the client can access the primary node without any problems.

7.3.4 Unplanned outages

The purpose of having HA systems is to minimize system downtime that is caused by unplanned outages, such as a power failure, a network failure, a DB2 failure, and so on. In this section, we simulate a DB2 system failure, a system crash, and a network failure to verify our setup, and show the transition states of the takeover process that Tivoli SA MP performs.

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

Run `db2_ki11` on the primary instance to stop the DB2 processes. Now all DB2 clients cannot connect to the database.

Run `lssam` to examine the resources. The HADR resource and the DB2 resource on the primary node are in the *Pending Online* state (Example 7-14).

Example 7-14 lssam output after you stop 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
```

Tivoli SA MP restarts the DB2 instance on the same node automatically. Tivoli SA MP does not execute takeover because the failed DB2 instance is recovered. All DB2 clients can connect to the database again.

Preventing a restart of the DB2 instance on primary node

In this test, we rename the DB2 start executable file so Tivoli SA MP cannot restart the DB2 instance. Run the following commands on the primary node to stop and break the DB2 instance:

```
(P) $mv ./sqllib/adm/db2star2 db2star2.mv ; db2_kill
```

The output of the **lssam** command shows that the HADR resource and the DB2 resource on the primary node are in the *Pending Online* state (Example 7-15). Now all DB2 clients cannot connect to the database.

Example 7-15 lssam output after you run db2_kill

```
(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
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:Balti
    c
  '- 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
```

Tivoli SA MP tries to start the DB2 instance but fails because the DB2 executable was renamed. Tivoli SA MP 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 (Example 7-16).

Example 7-16 lssam 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
Control=SuspendedPropagated
            |- 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
Control=SuspendedPropagated
            |- Offline IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
            '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

Tivoli SA MP continues trying to bring the DB2 instance resource online on what is now the old primary node. The timeout occurs 4 - 5 minutes afterward, and this *Pending Online* state is changed to the *Failed Offline* state (Example 7-17).

Example 7-17 lssam output after timeout

```
(P)(S) #lssam
Failed offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg
Control=MemberInProblemState 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
    Control=SuspendedPropagated
      |- 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
            Control=SuspendedPropagated
              |- 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, rename the **db2start** executable to its original name on the old primary node (Baltic) by running the following command:

```
(S) $mv ./db2star2.mv ./sql1ib/adm/db2star2
```

You must reset the HADR and DB2 resources on the old primary node to remove the Failed Offline flag. You accomplish this task by running **resetrsrc** commands (with root authority) in the following 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 7-18 shows the commands that we chose to run.

Example 7-18 Reset the resources

```
(P) (S) #CT_MANAGEMENT_SCOPE=2
(P) (S) #export CT_MANAGEMENT_SCOPE

(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 reaches the Peer state, the Lock status from the HADR resource group is removed. You can run these procedures without impacting the activities taking place at the new primary database and DB2 clients.

Primary node crash and reintegration of the old primary node

Here we describe node failure by simulating a node crash on the primary node. We simulate this crash by performing a power off or running `shutdown` on the node.

Node crash

Figure 7-6 illustrates the cluster behavior if there is a primary (service) node crash.

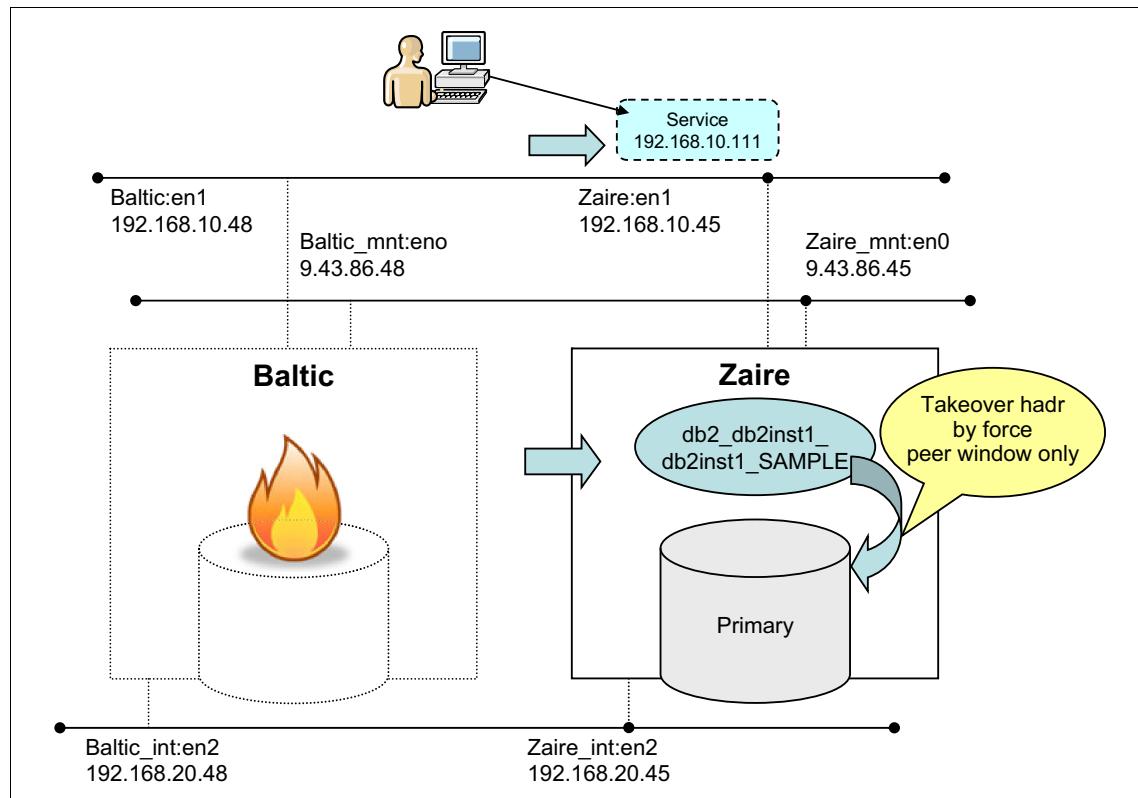


Figure 7-6 Automatic failover to a standby node

When the primary node crashes, the clients cannot connect to the database. Tivoli SA MP detects the primary node's outage and starts the failover process as follows:

1. The db2_db2inst1_db2inst1_SAMPLE-rg resource group 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 Tivoli SA MP, which is related to the resource group, is run on the standby node. The script includes the **TAKEOVER HADR** command, which changes the role of the standby database to primary.

After the failover, the resources are in the states that are shown in Example 7-19:

- ▶ The db2_db2inst1_db2inst1_SAMPLE-rg HADR resource group 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 7-19 Issam output after the primary node crash

```
(S) #lssam
Failed offline IBM.ResourceGroup:db2_db2inst1_Baltic_0-rg
Control=MemberInProblemState Nominal=Online
  '- Failed offline IBM.Application:db2_db2inst1_Baltic_0-rs
    Control=MemberInProblemState
      '- 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
    Control=SuspendedPropagated
      |- Failed offline
        IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Balti
        c Node=Offline
      '- Online
        IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
      '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
        Control=SuspendedPropagated
```

```
| - 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 in the *Lock* status after the failover. This lock indicates that the HADR databases are no longer in the 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 now has the SAMP resource group.

Reintegration

Figure 7-7 on page 262 shows the process that is required to reintegrate the old primary node in to clusters:

1. Baltic recovers from the crash.
2. When 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. When the HADR system reaches the Peer state, the lock from the HADR resource group is removed. Reintegration of the old primary database is complete.

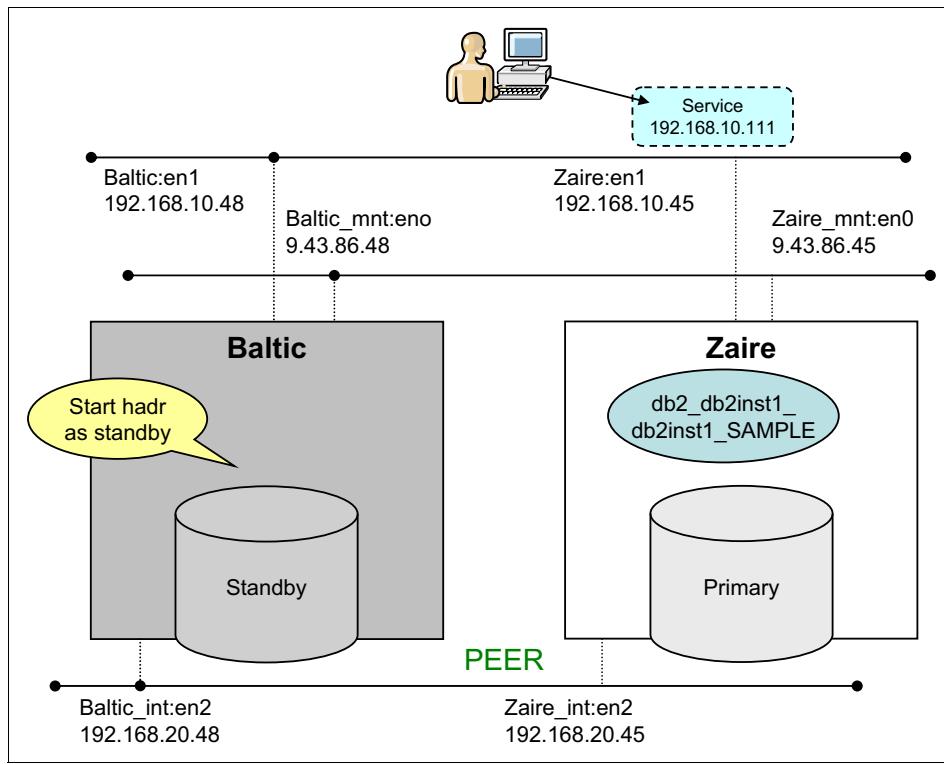


Figure 7-7 Reintegration of the old primary database

Network failure

Here we describe 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 (Baltic) to break the network, then run **lssam** to examine the state of the system resources. You can simulate the network failure by running **chdev**:

```
(P) # chdev -l 'en1' -a state='detach'
```

The HADR resource (**db2_db2inst1_db2inst1_SAMPLE-rg**) is in a Pending Online state (Example 7-20).

Example 7-20 lssam output after a network failure on the primary node

```
(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
PendingonlineIBM.ResourceGroup:db2_db2inst1_db2inst1_SAMPLE-rgControl=M
emberInProblemState Nominal=Online
        |- Pending online
            IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs
                |- Offline
                    IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Balti
c
                '- Pending online
                    IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Zaire
        '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
Control=MemberInProblemStated
        |- Failed offline
            IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
        '- Online IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire

```

Tivoli SA MP starts a failover operation. Run **1ssam** repeatedly to examine the progress of the failover. The system eventually enters one of 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 (Example 7-21).

Example 7-21 1ssam output after takeover

```

(P)(S) #1ssam
Failedoffline
IBM.ResourceGroup:db2_db2inst1_Baltic_0-rgControl=MemberInProblemSta
te 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
Control=SuspendedPropagated

```

```
| - Offline
    IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Ba
        ltic
    '- Online
        IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs:Za
            ire
'- Online IBM.ServiceIP:db2ip_192_168_10_111-rs
    Control=SuspendedPropagated
        |- 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 (Baltic). If you run **chdev** to simulate the failure, run this command again to change the state:

```
(S) # chdev -l 'en1' -a state='up'
```

For some environments, it might be necessary to restart the network interface. As root, run **smitty network** and select Minimum Configuration & Startup. Then, select en1 interface. Set yes as a value for the “START TCP/IP daemons Now” property and press Enter to run the selected command. Afterward, verify if network interface “en1” is listed as a resource on both nodes under equivalency for public network by running **lsequ** (Example 7-22).

Example 7-22 lsequ output after you restart the public network interface

```
(S) # lsequ -e db2_public_network_0
Displaying Equivalency information:
For Equivalency "db2_public_network_0".
```

Equivalency 1:

Name	= db2_public_network_0
MemberClass	= IBM.NetworkInterface
Resource:Node[Membership]	= { Unresolved ,en1:Zaire}
SelectString	= ""
SelectFromPolicy	= ANY
MinimumNecessary	= 1
Subscription	= {}
Color	= 0
ActivePeerDomain	= hadr_dom
ConfigValidity	=

If there is a “Unresolved” string in the “Resource:Node[Membership]” field, modify the existing configuration by running the following command:

```
(S) # chequ -u r db2_public_network_0  
IBM.NetworkInterface:en1:Baltic,en1:Zaire
```

In this case, the virtual IP resource on the primary node remains in the Failed Offline status. You must reset the virtual IP resource on the old primary node to remove the Failed Offline flag. Run the following commands 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
```

Before you run **resetrsrc**, ensure that the `CT_MANAGEMENT_SCOPE` environment variable is set to a proper value, for example:

```
(P) (S) #CT_MANAGEMENT_SCOPE=2  
(P) (S) #export CT_MANAGEMENT_SCOPE  
  
(P) (S) #resetrsrc -s 'Name like "db2ip_192_168_10_111-rs" &&  
NodeNameList={"Baltic"}' IBM.ServiceIP  
  
(P) (S) #resetrsrc -s "Name like 'db2_db2inst1_Baltic_0-rs'"  
IBM.Application
```

Again, run **lssam** repeatedly to examine the progress of the failover. The system eventually reaches the following states:

- ▶ The resource group of the old primary instance is Online.
- ▶ The old primary virtual IP resource remains Offline.

For more details, see Example 7-23.

Example 7-23 lssam output after you recover 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  
Request=Lock Nominal=Online  
  '- Online IBM.Application:db2_db2inst1_db2inst1_SAMPLE-rs C
```

```
| - 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
    Control=SuspendedPropagated
    | - Offline
      IBM.ServiceIP:db2ip_192_168_10_111-rs:Baltic
      '- Online
        IBM.ServiceIP:db2ip_192_168_10_111-rs:Zaire
```

7.4 DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system

Here we present a test case that combines the DB2 HA features to automate failover of HADR databases on Linux. It shows how **db2haicu** interactive mode can be used set up the Tivoli SA MP.

7.4.1 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 11 SP1 for AMD64/Intel64
- ▶ One network interface card per server
- ▶ Common network access to a third gateway device as a tiebreaker
- ▶ DB2 10.1 Advanced Enterprise Server Edition with fix pack 1 and Tivoli SA MP V3.2 installed
- ▶ HADR set up on a database

Table 7-1 shows the naming conventions that we used in this example.

Table 7-1 DB2 HADR naming

Server	lepus	mensa
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: subnet mask	9.43.86.252: 255.255.252.0	9.43.86.252: 255.255.252.0

HADR and the service port number: 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.

7.4.2 Configuration

In this section, we provide individual configuration steps, followed by the execution of the **db2haicu** utility to create a cluster. The steps and tasks that are involved in setting up a cluster that is integrated with DB2 High Availability Feature are covered in detail in the section “Configuring for high availability”, in Chapter 4, “Configuring for high availability”, in *Data Recovery and High Availability Guide and Reference*, SC27-3870-00.

Before you run db2haicu

Based on the provided architecture for our example, complete the following configuration steps before you run **db2haicu**:

1. Ensure that the matching entries for DB2 instance communication and HADR ports exist in the /etc/services files for both servers that are added as cluster nodes. Example 7-24 shows the entries for our configuration.

Example 7-24 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 the /etc/hosts files for both servers that are added as cluster nodes. Example 7-25 shows the entries for our configuration.

Example 7-25 Both host name 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 that is being used to create the cluster domain. This command has most likely been added to the `~/.profile` of the user ID by the **db2icrt** command.
4. Start the database manager on both servers by running **db2start**.
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 the Peer state.
6. Set the 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 the instance owner ID:
`db2 update db cfg for sample using HADR_PEER_WINDOW 120`

7. To prepare each node for clustering:

On both servers, run **preprnode** 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 run.

The syntax for the **preprnode** command is covered in *RSCT for Multiplatforms: Technical Reference*, SA22-7893-19 or in the RSCT Information Center at:

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.v3r2.rsct700.doc/b1501m_about.htm

RSCT on virtual machines

If the environment is running on cloned virtual images (for example, VMware), then the RSCT node identifier is not unique, which prevents the 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. For more information, see the Information Center at:

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.rsct.v3r2.rsct100.doc/b1503_about.htm

This error can be corrected, if there is no cluster present, by running the following command:

```
/usr/sbin/rsct/install/bin/recfgct
```

This command effectively reinitializes the RSCT base, and one feature is to give each node a unique identifier.

Cluster definitions: If there is any cluster definition present when the **recfgct** command is run, the effects are unpredictable, and symptoms might include immediate recycling of the operating system.

The db2haicu configuration

At this stage, the environment should be ready for the **db2haicu** command. It should be first run on the HADR standby node, and then on the HADR primary node to successfully complete cluster definition with HADR integration.

When every prerequisite is met, the assumptions that are made by **db2haicu** allow usage of the default response to almost each prompt. We use the defaults wherever possible, *emphasizing* places where we made specific entries.

Example 7-26 shows the initial execution of **db2haicu** on our HADR standby node after you confirm that HADR is running in the Peer state as the standby.

Example 7-26 db2haicu initial pass on HADR standby node

```
db2inst2@mensa:~> db2pd -d sample -hadr

Database Member 0 -- Database SAMPLE -- Standby -- Up 2 days 19:03:16
-- Date 2012-08-06-05.37.11.820815

          HADR_ROLE = STANDBY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = SYNC
          STANDBY_ID = 0
          LOG_STREAM_ID = 0
          HADR_STATE = PEER
          PRIMARY_MEMBER_HOST = lepus
          PRIMARY_INSTANCE = db2inst2
          PRIMARY_MEMBER = 0
          STANDBY_MEMBER_HOST = mensa
          STANDBY_INSTANCE = db2inst2
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 08/06/2012 05:30:16.969954
(1344245416)
          HEARTBEAT_INTERVAL(seconds) = 1
          HADR_TIMEOUT(seconds) = 3
          TIME_SINCE_LAST_RECV(seconds) = 0
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
          LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.003236
          LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.142
          LOG_HADR_WAIT_COUNT = 44
          SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
          SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
          PRIMARY_LOG_FILE,PAGE,POS = S0000039.LOG, 260, 200785896
          STANDBY_LOG_FILE,PAGE,POS = S0000039.LOG, 260, 200785896
          HADR_LOG_GAP(bytes) = 0
          STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000039.LOG, 260, 200785896
          STANDBY_RECV_REPLY_GAP(bytes) = 0
          PRIMARY_LOG_TIME = 08/06/2012 05:30:14.000000
(1344245414)
```

```
        STANDBY_LOG_TIME = 08/06/2012 05:30:14.000000
(1344245414)      STANDBY_REPLY_LOG_TIME = 08/06/2012 05:30:14.000000
(1344245414)
    STANDBY_RECV_BUF_SIZE(pages) = 512
    STANDBY_RECV_BUF_PERCENT = 0
    STANDBY_SPOOL_LIMIT(pages) = 0
    PEER_WINDOW(seconds) = 120
    PEER_WINDOW_END = 08/06/2012 05:39:11.000000
(1344245951)
    READS_ON_STANDBY_ENABLED = N
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 apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu 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 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 exit.

Create a domain and continue? [1]

1. Yes

2. No

Create a unique name for the new domain:

lepusmensa

Nodes must now be added to the new domain.

How many cluster nodes the domain 'lepusmensa' 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 exit.

Create the domain now? [1]

1. Yes

2. No

Creating domain 'lepusmensa' in the cluster ...

Creating domain 'lepusmensa' in the cluster was successful.

Interrupt: 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 run **db2haicu** again, which skips over the next stage of creating a quorum device, or you can run **db2haicu -delete** and then try running **db2haicu** again. You can add a quorum device later through the **db2haicu** maintenance mode menu item 6 "Create a new quorum device for the domain". Run **db2haicu -delete** because it ensures 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 have to manually intervene if subsets of machines in the cluster lose connectivity.

Configure a quorum device for the domain called 'lepusmensa'? [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 the following total number of network  
interface cards on the machines in the cluster domain: '2'. You can  
add a network to your cluster domain using the db2haicu utility.  
  
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' ...
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 ...
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
existing ...
db2inst2@mensa:~>
```

This process brings us *almost* to the state of a working cluster. The HADR database pair is not added to the cluster yet because the database on the current cluster node is in the HADR standby role. **db2haicu** must now be run on the primary node to add the SAMPLE database for both nodes in to the cluster.

On a clean system with no residual cluster definitions, the output of **db2haicu** in *startup mode* resembles Example 7-27. This example 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 choose to use a virtual IP address in our test environment, with our remote client running **db2 catalog tcpip node** commands to point at that virtual IP address.

Example 7-27 db2haicu on HADR Primary node - first run on a clean system

```
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 apply to this instance.

`db2haicu` is collecting information on your current setup. This step may take some time as `db2haicu` 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 '`lepusmensa`' on this machine. The cluster configuration that follows 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

2. db2_private_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:~>

We now have a fully completed and integrated HADR cluster. You can see what cluster objects are defined by running the Tivoli SA MP **1ssam** command (Example 7-28). This output shows us:

- ▶ The resource group for the HADR database pair (SAMPLE on lepus, as the HADR primary, is shown as “Online”, while SAMPLE on mensa, as the HADR standby, is shown as “Offline”).
- ▶ The Application definition of the virtual service IP address (again, lepus, as the HADR primary, is shown as “Online”, while mensa, as the HADR standby, is shown as “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 7-28 Tivoli SA MP view of cluster objects that are created by db2haicu

```
db2inst2@lepus:~> 1ssam
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
```

7.4.3 Testing

Here we describe the steps of a standard set of tests to prove that the cluster controlled by the DB2 High Availability Feature can handle common failure scenarios in an appropriate manner. These scenarios 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 the Peer state, that our HADR Primary role is held by lepus, and HADR standby is mensa, as shown by the output of **lssam** and **db2pd** commands in Example 7-29.

Example 7-29 The assumed starting point for our testing scenarios

```
db2inst2@lepus:~> 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: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

db2inst2@lepus:~> db2pd -d sample -hadr
```

```
Database Member 0 -- Database SAMPLE -- Active -- Up 2 days 20:52:44 --
Date 2012-08-06-07.26.48.066630
```

```
HADR_ROLE = PRIMARY
REPLAY_TYPE = PHYSICAL
HADR_SYNCMODE = SYNC
STANDBY_ID = 1
LOG_STREAM_ID = 0
HADR_STATE = PEER
PRIMARY_MEMBER_HOST = lepus
PRIMARY_INSTANCE = db2inst2
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = mensa
STANDBY_INSTANCE = db2inst2
STANDBY_MEMBER = 0
HADR_CONNECT_STATUS = CONNECTED
```

```

        HADR_CONNECT_STATUS_TIME = 08/06/2012 05:30:16.970056
(1344245416)
        HEARTBEAT_INTERVAL(seconds) = 1
        HADR_TIMEOUT(seconds) = 3
        TIME_SINCE_LAST_RECV(seconds) = 1
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.002415
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.713
        LOG_HADR_WAIT_COUNT = 287
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200983920
        STANDBY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200983920
        HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200983920
        STANDBY_RECV_REPLY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 08/06/2012 07:18:53.000000
(1344251933)
        STANDBY_LOG_TIME = 08/06/2012 07:18:53.000000
(1344251933)
        STANDBY_REPLY_LOG_TIME = 08/06/2012 07:18:53.000000
(1344251933)
        STANDBY_RECV_BUF_SIZE(pages) = 512
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 0
        PEER_WINDOW(seconds) = 120
        PEER_WINDOW_END = 08/06/2012 07:28:47.000000
(1344252527)
        READS_ON_STANDBY_ENABLED = N

```

Monitoring scripts

During testing, we monitor the status of the cluster with a single-line shell script that issues `lssam` repeatedly, echoes an empty line, and sleeps for 5 seconds:

```
while : ;do lssam;echo "";sleep 5;done
```

We also monitor and 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 definitions are run 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**

We use the Virtual IP address that is 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 High Availability 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 run **chrg** or the cluster manager command that is normally required to make a change in the cluster, you can run **db2stop**, **db2start**, **stop hadr**, **start hadr**, and **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 run the **1ssam** command to monitor the current Online/Offline state of cluster resources. There is no requirement to **su** to the root user.

A complete list of the DB2 native commands that are integrated with the cluster manager can be found in the DB2 10.1 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.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 (the current HADR standby) run the following command:

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

The step-by-step results for our first **takeover** command are shown in Example 7-30 on page 281 to Example 7-35 on page 283.

The first change is a Request=Lock state on the HADR resource group (Example 7-30).

Example 7-30 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: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
```

The next noticeable change is a Request=Move, and the virtual IP address is set Offline (Example 7-31).

Example 7-31 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:mena
    '- 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: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
```

In Example 7-32, the HADR resource for the primary is set to Pending Offline.

Example 7-32 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:mena
    '- 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: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
```

The virtual IP address for mensa is set to Online, and the HADR database resource for mensa is Pending Online (Example 7-33).

Example 7-33 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:mena
    '- 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: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
```

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 (Example 7-34).

Example 7-34 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
```

Control of the DB2 command line returns to the administrator. The SAMPLE database on lepus now has the HADR standby role (Example 7-35), which is confirmed by running **db2pd**. With the SAMPLE database on mensa having the HADR primary role, the Service IP address resource is assigned to mensa.

Example 7-35 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 Member 0 -- Database SAMPLE -- Active -- Up 2 days 21:51:16 --
Date 2012-08-06-08.25.11.518820

          HADR_ROLE = PRIMARY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = SYNC
          STANDBY_ID = 1
          LOG_STREAM_ID = 0
          HADR_STATE = PEER
PRIMARY_MEMBER_HOST = mensa
          PRIMARY_INSTANCE = db2inst2
          PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = lepus
          STANDBY_INSTANCE = db2inst2
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 08/06/2012 05:30:16.969954

(1344245416)
          HEARTBEAT_INTERVAL(seconds) = 1
          HADR_TIMEOUT(seconds) = 3
          TIME_SINCE_LAST_RECV(seconds) = 0
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
          LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
          LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
          LOG_HADR_WAIT_COUNT = 0
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
          PRIMARY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200986141
          STANDBY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200986141
          HADR_LOG_GAP(bytes) = 0
          STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000039.LOG, 309, 200986141
          STANDBY_RECV_REPLY_GAP(bytes) = 0
          PRIMARY_LOG_TIME = 08/06/2012 07:29:58.000000

(1344252598)
          STANDBY_LOG_TIME = 08/06/2012 07:29:58.000000
(1344252598)
          STANDBY_REPLY_LOG_TIME = 08/06/2012 07:29:58.000000
(1344252598)
          STANDBY_RECV_BUF_SIZE(pages) = 512
          STANDBY_RECV_BUF_PERCENT = 0
          STANDBY_SPOOL_LIMIT(pages) = 0
          PEER_WINDOW(seconds) = 120

```

```
PEER_WINDOW_END = 08/06/2012 08:27:10.000000
(1344256030)
READS_ON_STANDBY_ENABLED = N
```

The output from our looping remote DB2 client script in Example 7-36 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 7-36 Brief outage of remote DB2 connectivity during takeover

Database Connection Information

```
Database server      = DB2/LINUXX8664 10.1.1
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/LINUXX8664 10.1.1
SQL authorization ID = DB2INST2
```

```
Local database alias = TESTSAMP
```

```
DB20000I The SQL command completed successfully.
```

The only reason that we are even seeing the SQL1776N messages in Example 7-36 on page 285 is that for this particular test environment we did not configure DB2 automatic client reroute (ACR) between the two HADR databases. This setup provides a useful contrast with the output from testing that we perform in 7.4, “DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system” on page 266, where we configure ACR in addition to the Virtual IP resource. Use ACR in addition to a cluster managed virtual IP address, as a useful means of avoiding remote DB2 connections that point at an HADR standby database.

Return the HADR Primary role back to the SAMPLE database on lepus by running the following command as the db2inst2 user on lepus:

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

The process of unforced HADR takeover from mensa back to lepus shows much the same output when you run **1ssam**, only with the other resources in a given resource group pair doing the Pending Offline, Offline, Pending Online, and Online status changes.

Unplanned failure of the HADR primary DB2 instance

Because the DB2 High Availability Feature integrated the **db2stop** command into cluster management, you must use a technique that effectively stops the DB2 instance outside of the cluster manager.

The **db2_kill** command is an excellent way to achieve this task. To test it, start the looping **1ssam** monitor script on either one of the cluster nodes, and the IBM DB2 Connect™ script on the remote client, then have the DB2 instance user perform the **db2_kill** command on the HADR primary node lepus, and capture the results.

The **db2_kill** command works as shown in Example 7-37.

Example 7-37 db2_kill performs its function

```
db2inst2@lepus:~> db2_kill
ipclean: Removing DB2 engine and client's IPC resources for db2inst2.
```

The looping `lssam` script output from Example 7-38 shows that the first preferred action of the cluster manager in this situation is to simply attempt to restart the DB2 instance that failed. This situation exemplifies the importance of ensuring that if you want to make DB2 resources unavailable for maintenance purposes, use commands that the cluster manager is aware of so that it does not automatically attempt to override your actions.

Example 7-38 shows 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 7-38 Unplanned failure - lssam 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: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
    .
    .
    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
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:mena
```

Example 7-39 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 lapse is picked up in the next iteration, where the HADR primary database resource on lepus is shown with a state of Unknown. This situation is where the HADR Primary database is not yet activated, and DB2 must perform crash recovery.

Example 7-39 Unplanned failure - lssam 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: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
  .
  .
  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: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
  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:mena
```

Example 7-40 shows the final range of actions that are 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 7-40 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 did not enable ACR, this output also indicates 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 action matches the behavior of the Tivoli SA MP cluster we test in 7.4, “DB2 HADR with Tivoli SA MP configuration for automatic failover on a Linux system” on page 266.

When the cluster returns all resources to their nominal state, run **db2pd** (Example 7-41), which confirms that the SAMPLE database on lepus retains the HADR Primary role.

Example 7-41 Return of the HADR primary

```
db2inst2@lepus:~> db2pd -d sample -hadr
```

```
Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:02:07 --
Date 2012-08-06-12.54.52.362268
```

```
          HADR_ROLE = PRIMARY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = SYNC
          STANDBY_ID = 1
          LOG_STREAM_ID = 0
          HADR_STATE = PEER
          PRIMARY_MEMBER_HOST = lepus
          PRIMARY_INSTANCE = db2inst2
          PRIMARY_MEMBER = 0
          STANDBY_MEMBER_HOST = mensa
          STANDBY_INSTANCE = db2inst2
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 08/06/2012 12:52:49.336918
(1344271969)
          HEARTBEAT_INTERVAL(seconds) = 1
          HADR_TIMEOUT(seconds) = 3
          TIME_SINCE_LAST_RECV(seconds) = 0
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
          LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
          LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
          LOG_HADR_WAIT_COUNT = 0
          SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
          SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
```

```

        PRIMARY_LOG_FILE,PAGE,POS = S0000039.LOG, 402, 201366148
        STANDBY_LOG_FILE,PAGE,POS = S0000039.LOG, 402, 201366148
            HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000039.LOG, 402, 201366148
            STANDBY_RECV_RECV_GAP(bytes) = 0
                PRIMARY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
                STANDBY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
                STANDBY_RECV_RECV_GAP(bytes) = 0
(1344266933)
                STANDBY_RECV_BUF_SIZE(pages) = 512
                    STANDBY_RECV_BUF_PERCENT = 0
                    STANDBY_SPOOL_LIMIT(pages) = 0
                        PEER_WINDOW(seconds) = 120
                            PEER_WINDOW_END = 08/06/2012 12:56:51.000000
(1344272211)
                READS_ON_STANDBY_ENABLED = N

```

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 reintegrated back into the cluster, and into the HADR database pair.

After you start the remote DB2 Connect script on the client, and the `1ssam` 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 task is to run `shutdown -h now` as the root user.

The output from our `1ssam` script in Example 7-42 on page 292 and Example 7-43 on page 293 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. Now our lepus node is powered off.

Example 7-42 shows the cluster manager is 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 7-42 Cluster manager is setting the resources for our lepus node to Offline and Failed Offline

```
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
  .
  .
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:mensa
  '- 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:mensa
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:mensa
```

Example 7-43 shows that the cluster manager is progressing from noticing that the primary node is 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=MemberInProblemState** flag that is issued against the DB2 instance resource group for lepus. This flag protects the integrity of the cluster if lepus comes online again. Before lepus allows any external connectivity, it must perform the appropriate cluster actions to satisfy the requirements to overcome the **Control=MemberInProblemState** flag.

Example 7-43 Cluster manager notices that the primary node is in the Failed Offline state

```
Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg
Control=MemberInProblemState Nominal=Online
  |- Pending online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=MemberInProblemState
  |- 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=MemberInProblemState
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 Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- 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 Control=SuspendedPropagated
    |- 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=MemberInProblemState
Nominal=Online
  '- Failed offline IBM.Application:db2_db2inst2_lepus_0-rs
Control=MemberInProblemState
    '- 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 7-44 shows no point where DB2 displays lost connectivity. A forced takeover and Virtual IP switch that is managed by our cluster manager are effective and rapid.

Example 7-44 DB2 Connect output

```
db2inst1@itso:~/script> ./conntest.sh
```

```
Database Connection Information
```

```
Database server      = DB2/LINUXX8664 10.1.1
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 reintegrates into the cluster.

Example 7-45 shows when the cluster manager detects that the lepus node is back and available for cluster activity.

Example 7-45 The lepus node is available for cluster activity

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- 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 Control=SuspendedPropagated
    |- 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=MemberInProblemState
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:mena
  .
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mena
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs Control=SuspendedPropagated
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:mena
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:mena

```

Example 7-46 shows that the lepus node is now fully reintegrated in to the cluster with a DB2 HADR standby role. The DB2 instance resource group for lepus is set to Online, the DB2 HADR database resource for lepus is set to Offline, and SuspendedPropagated and Lock flags are removed, meaning that the lepus node successfully started as an HADR Standby database.

Example 7-46 The lepus node is fully reintegrated and HADR is standing by

```

Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:lepus
  '- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs:mena
  '- Online IBM.ServiceIP:db2ip_192_168_145_252-rs Control=SuspendedPropagated
    |- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Online 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

```

```

. .
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:mena
  '- 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: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

```

For DB2 Connect output, see Example 7-44 on page 294. DB2 connectivity is never lost. Example 7-47 shows that DB2 HADR is connected and in the Peer state, and that the SAMPLE database has the standby role on lepus.

Example 7-47 SAMPLE database is on standby on lepus

```
db2inst2@lepus:~> db2pd -d sample -hadr
```

```
Database Member 0 -- Database SAMPLE -- Standby -- Up 0 days 00:11:16
-- Date 2012-08-06-13.32.16.605775
```

```

          HADR_ROLE = STANDBY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = SYNC
          STANDBY_ID = 0
          LOG_STREAM_ID = 0
          HADR_STATE = PEER
          PRIMARY_MEMBER_HOST = mensa
          PRIMARY_INSTANCE = db2inst2
          PRIMARY_MEMBER = 0
          STANDBY_MEMBER_HOST = lepus
          STANDBY_INSTANCE = db2inst2
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 08/06/2012 13:21:00.654849
(1344273660)
          HEARTBEAT_INTERVAL(seconds) = 1
          HADR_TIMEOUT(seconds) = 3
          TIME_SINCE_LAST_RECV(seconds) = 0

```

```

        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.002259
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.998
                    LOG_HADR_WAIT_COUNT = 470
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
                PRIMARY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
                STANDBY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
                    HADR_LOG_GAP(bytes) = 0
STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
                STANDBY_RECV_REPLY_GAP(bytes) = 0
                    PRIMARY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)                      STANDBY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)                      STANDBY_REPLY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
                STANDBY_RECV_BUF_SIZE(pages) = 512
                STANDBY_RECV_BUF_PERCENT = 0
                STANDBY_SPOOL_LIMIT(pages) = 0
                    PEER_WINDOW(seconds) = 120
                    PEER_WINDOW_END = 08/06/2012 13:34:14.000000
(1344274454)
                READS_ON_STANDBY_ENABLED = N

```

Our final step to bring our cluster back to its initial state by running an unforced HADR takeover command from lepus. The results are shown in the **db2pd** output in Example 7-48. The output from our remote DB2 Connect script during this process mirrors Example 7-36 on page 285 exactly.

Example 7-48 Bringing a cluster back to its initial state through an unforced takeover

```

db2inst2@lepus:~> db2 takeover hadr on db sample
DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.
db2inst2@lepus:~> db2pd -d sample -hadr

```

```

Database Member 0 -- Database SAMPLE -- Active -- Up 0 days 00:12:40 --
Date 2012-08-06-13.33.40.774711

```

```

        HADR_ROLE = PRIMARY
        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = SYNC
        STANDBY_ID = 1

```

```

LOG_STREAM_ID = 0
HADR_STATE = PEER
PRIMARY_MEMBER_HOST = lepus
PRIMARY_INSTANCE = db2inst2
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = mensa
STANDBY_INSTANCE = db2inst2
STANDBY_MEMBER = 0
HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 08/06/2012 13:21:00.654849
(1344273660)
    HEARTBEAT_INTERVAL(seconds) = 1
    HADR_TIMEOUT(seconds) = 3
    TIME_SINCE_LAST_RECV(seconds) = 1
    PEER_WAIT_LIMIT(seconds) = 0
    LOG_HADR_WAIT_CUR(seconds) = 0.000
    LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
    LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
    LOG_HADR_WAIT_COUNT = 0
    SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
    SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
        STANDBY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
        HADR_LOG_GAP(bytes) = 0
    STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
    STANDBY_RECV_REPLY_GAP(bytes) = 0
    PRIMARY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
    STANDBY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
    STANDBY_REPLY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
    STANDBY_RECV_BUF_SIZE(pages) = 512
    STANDBY_RECV_BUF_PERCENT = 0
    STANDBY_SPOOL_LIMIT(pages) = 0
    PEER_WINDOW(seconds) = 120
    PEER_WINDOW_END = 08/06/2012 13:35:40.000000
(1344274540)
    READS_ON_STANDBY_ENABLED = N

```

This concludes our testing of HADR integrated into a managed cluster with the DB2HA feature.

7.4.4 Administration

Here we provide a brief overview about how to perform maintenance activities on DB2 resources without the cluster manager attempting to restart them when they need to be offline.

We mentioned in “Planned HADR takeover” on page 280 how DB2 commands are now integrated into the cluster manager. This integration makes administration and maintenance of DB2 objects intuitive.

For our examples here, we use the same test environment and initial cluster state that is used in 7.4.3, “Testing” on page 277.

Our first example is to run **db2stop** on the DB2 instance with the HADR standby database. We measure the results using our looping **1ssam** script from “Monitoring scripts” on page 279.

Example 7-49 shows the output from the **db2stop** command. We must run a DB2 **deactivate database** command against an HADR standby database.

Example 7-49 db2stop on the HADR standby instance

```
db2inst2@mensa:~> db2 deactivate db sample
DB20000I  The DEACTIVATE DATABASE command completed successfully.
db2inst2@mensa:~> db2stop
08/06/2012 13:47:39 0 0  SQL1064N  DB2STOP processing was
successful.
SQL1064N  DB2STOP processing was successful.
```

Example 7-50 shows the **1ssam** 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 7-50 DB2 stopped with the DB2 High Availability 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:mena
  .
  .
  Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
    |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- 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 Control=SuspendedPropagated
    |- 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
Pendingonline IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Request=Lock Nominal=Online
  '- Offline IBM.Application:db2_db2inst2_mensa_0-rs
Control=StartInhibitedBecauseSuspended
  '- Offline IBM.Application:db2_db2inst2_mensa_0-rs:mena

```

Example 7-51 shows that we are manually restarting the DB2 instance, and the output of a **db2pd** command confirms that the HADR standby database is automatically activated. We perform the **db2pd** test primarily because it is difficult to tell from output of the **lssam** command whether the HADR resource is in standby or is offline.

Example 7-51 DB2 starts and the HADR standby is activated automatically

```

db2inst2@mena:~> db2start
08/06/2012 13:50:01 0 0  SQL1063N  DB2START processing was
successful.
SQL1063N  DB2START processing was successful.
db2inst2@mena:~> db2 start hadr on db sample as standby
DB20000I  The START HADR ON DATABASE command completed successfully.
db2inst2@mena:~> db2pd -d sample -hadr

```

```

Database Member 0 -- Database SAMPLE -- Standby -- Up 0 days 00:00:22
-- Date 2012-08-06-13.42.47.375474

```

```

          HADR_ROLE = STANDBY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = SYNC
          STANDBY_ID = 0
          LOG_STREAM_ID = 0

```

```

          HADR_STATE = PEER
PRIMARY_MEMBER_HOST = lepus
          PRIMARY_INSTANCE = db2inst1
          PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = mensa
          STANDBY_INSTANCE = db2inst1
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 08/06/2012 13:42:26.923441
(1344274946)
          HEARTBEAT_INTERVAL(seconds) = 1
          HADR_TIMEOUT(seconds) = 3
          TIME_SINCE_LAST_RECV(seconds) = 0
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
          LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
          LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
          LOG_HADR_WAIT_COUNT = 0
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
          PRIMARY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
          STANDBY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
          HADR_LOG_GAP(bytes) = 0
          STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000040.LOG, 0, 203800046
          STANDBY_RECV_REPLY_GAP(bytes) = 0
          PRIMARY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
          STANDBY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
          STANDBY_REPLY_LOG_TIME = 08/06/2012 11:28:53.000000
(1344266933)
          STANDBY_RECV_BUF_SIZE(pages) = 512
          STANDBY_RECV_BUF_PERCENT = 0
          STANDBY_SPOOL_LIMIT(pages) = 0
          PEER_WINDOW(seconds) = 120
          PEER_WINDOW_END = 08/06/2012 13:44:48.000000
(1344275088)
          READS_ON_STANDBY_ENABLED = N

```

Example 7-52 shows the **lssam** output with a simple transition between the DB2 instance that is being stopped (Offline), and restarting it to be automatically reintegrated into the cluster (Online). According to the cluster manager, there is no change in state for the HADR standby database resource, although we know that it must track internally 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 7-52 Reintegration is smooth with the DB2 High Availability Feature

```
Online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock Nominal=Online
  |- Online IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=SuspendedPropagated
  |- 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 Control=SuspendedPropagated
    |- 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
Pendingonline IBM.ResourceGroup:db2_db2inst2_mensa_0-rg Request=Lock Nominal=Online
  '- Offline IBM.Application:db2_db2inst2_mensa_0-rs
Control=StartInhibitedBecauseSuspended
  '- 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 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 is stopped to perform administration or maintenance activities.

You see in “Unplanned failure of the HADR primary DB2 instance” on page 286 how the cluster manager reacts to a **db2 kill** command issued outside its control. For our next example, we attempt to run **db2stop** against the DB2 instance with the HADR primary database.

Example 7-53 is the output from our attempt to stop a DB2 HADR primary database and instance.

Example 7-53 Stopping an HADR primary resource

```
db2inst2@lepus:~> db2 list applications
SQL1611W No data was returned by Database System Monitor.
db2inst2@lepus:~> db2 deactivate db sample
DB20000I The DEACTIVATE DATABASE command completed successfully.
db2inst2@lepus:~> db2stop
08/06/2012 13:54:05 0 0 SQL1064N DB2STOP processing was
successful.
SQL1064N DB2STOP processing was successful.
```

Example 7-54 to Example 7-56 on page 305 shows what we suspected might happen. The cluster acts to restart the DB2 instance and reactivate the HADR primary database.

Example 7-54 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: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
...
...
```

```

Pending online IBM.ResourceGroup:db2_db2inst2_db2inst2_SAMPLE-rg Request=Lock
Nominal=Online
  |- Offline IBM.Application:db2_db2inst2_db2inst2_SAMPLE-rs
Control=StartInhibitedBecauseSuspended
  |- Offline 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 Control=SuspendedPropagated
    |- Online IBM.ServiceIP:db2ip_192_168_145_252-rs:lepus
    '- Offline IBM.ServiceIP:db2ip_192_168_145_252-rs:mena
Pending online IBM.ResourceGroup:db2_db2inst2_lepus_0-rg Request=Lock Nominal=Online
  '- Offline IBM.Application:db2_db2inst2_lepus_0-rs
Control=StartInhibitedBecauseSuspended
  '- 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:mena

```

Example 7-55 shows the progression from the DB2 instance that is restarted and set back to the Online state, to where the HADR database resource for lepus is optimistically set to Pending Online.

Example 7-55 Stopping HADR primary - part 2 of 3

```

db2inst2@lepus:~> db2 activate db sample
DB20000I  The ACTIVATE DATABASE command completed successfully.
db2inst2@lepus:~> lssam
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: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
  ..
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: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
```

In Example 7-56, we see that the cluster reports that the HADR database resource group and resource for lepus are set to Online, and all is back to nominal.

Example 7-56 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: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
```

7.5 Automating HADR takeover with PowerHA

This section describes how to automate DB2 HADR takeover in an PowerHA environment. The section also describes the failover test procedure for HADR controlled by the PowerHA facility.

PowerHA for AIX provides a highly available computing environment. PowerHA 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 PowerHA clusters is to provide a highly available environment for mission-critical applications.

In a PowerHA cluster, to ensure the availability of these applications, the applications are placed under PowerHA control. PowerHA ensures that the applications remain available even when a component in a cluster fails. To ensure availability if there is a component failure, the PowerHA software moves the application, along with the resources, to another node in the cluster. For more information about PowerHA, see Chapter 3, “DB2 and PowerHA SystemMirror” on page 71.

7.5.1 PowerHA and HADR planning

Before you set up an HADR and PowerHA environment, you must plan the cluster environment. The following items must be considered:

- ▶ Physical nodes
- ▶ Network
- ▶ PowerHA configuration
- ▶ HADR configuration
- ▶ Scripts

Lab environment

Figure 7-8 shows the configuration of PowerHA and HADR in our lab environment.

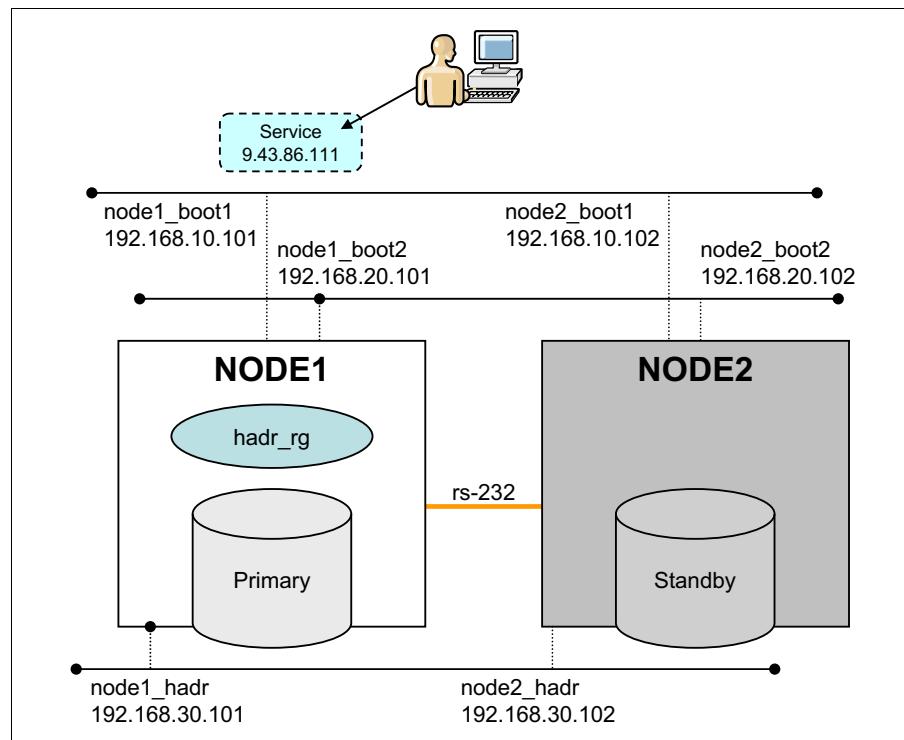


Figure 7-8 Lab environment

Our planned 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, where the HADR standby database is. The role of both of these nodes changes in response to system failover or planned takeover that is 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 PowerHA. The service address for the clients is added on one of these network interfaces.

- One Ethernet network interface is dedicated to HADR communications. Have a separate network for HADR if you want to avoid interference by HADR log transfer.
- One serial (RS-232C) network is configured for PowerHA keep-alive. Having a serial network (non-TCP/IP network) makes PowerHA failure detection more reliable.

Shared disks: It is not necessary to have shared disks for HADR in an PowerHA cluster because the primary and standby databases are independent databases, which can be in separate storage devices.

- ▶ PowerHA 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 PowerHA resource group. IP address takeover is optional if you use ACR, where clients can automatically switch the nodes that they connect to.
- ▶ HADR configuration:
 - Each node has the DB2 instance named hadrinst. 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 a primary node crash

Figure 7-9 shows the steps that the cluster performs if there is a primary (service) node crash:

1. PowerHA 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 that is related to the resource group is added to the standby node.
3. The Start script in the PowerHA 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 PowerHA resource group now.

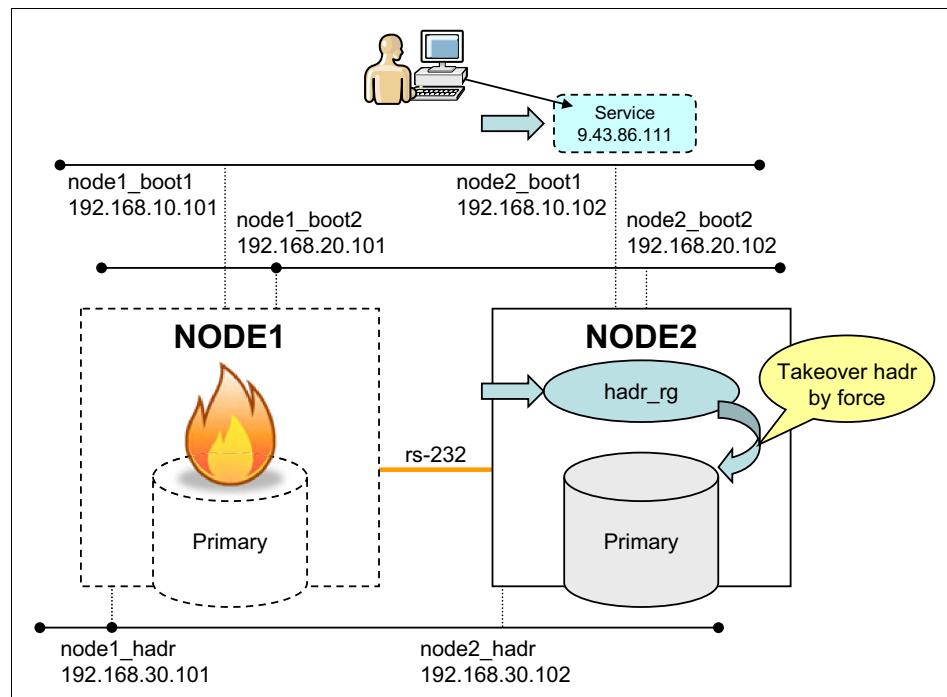


Figure 7-9 Automatic failover to standby node

Reintegration of the old primary node

Figure 7-10 shows the process that is required to reintegrate the old primary node into clusters:

1. After Node1 recovers from a hardware crash, you can start the instance and start HADR as the standby database on this node by running `start hadr` with the `as standby` option.
2. The standby 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.
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 is complete.

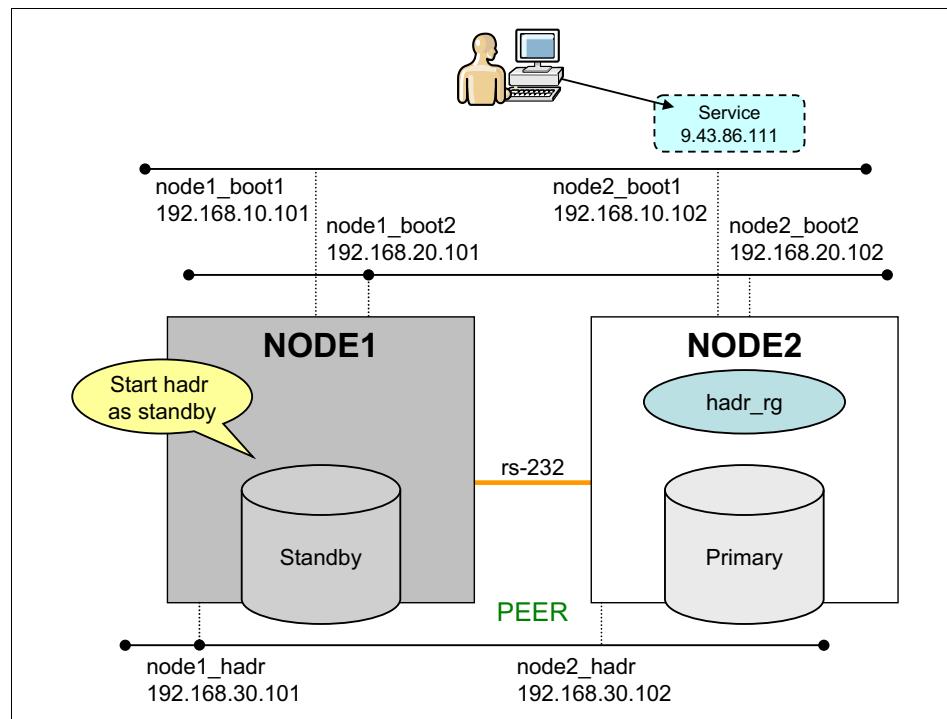


Figure 7-10 Reintegration of the old primary database

7.5.2 Step-by-step configuration overview

You can configure automated HADR takeover environment with PowerHA by completing the following operations:

1. Perform the HADR setup.

2. Perform the PowerHA setup. Here we test only the PowerHA configuration and function with dummy scripts.
3. Prepare the scripts that are required to control the HADR database from PowerHA.
4. Perform a joint test of HADR and PowerHA:
 - Normal start of cluster
 - Planned takeover
 - Unplanned takeover
 - Normal stop of cluster

7.5.3 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 hadrininst.
2. Check that the correct entries are configured in the /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 the HADR parameters correctly on both 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.

For more information about the step-by-step configuration of HADR, see Chapter 6, “HADR setup” on page 173.

Figure 7-11 shows the entries for the services and hosts of the HADR configuration in our lab environment.

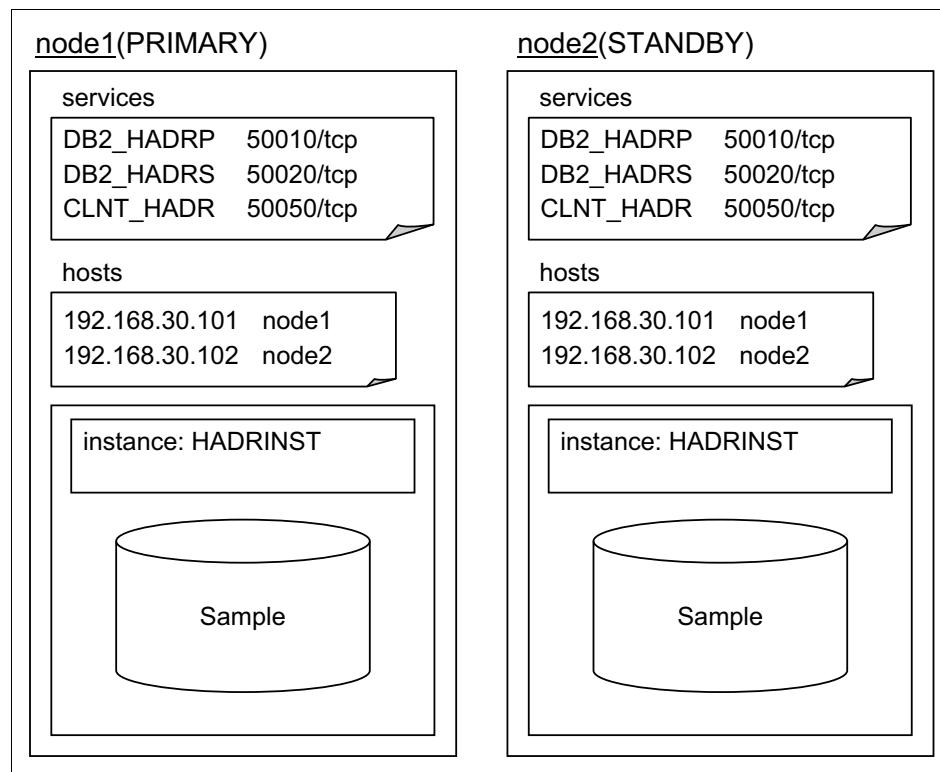


Figure 7-11 Instance, services, and hosts entries for HADR configuration

7.5.4 PowerHA configuration

Here we describe the procedure to configure an PowerHA cluster for automatic failure detection of the primary node.

Checking the Internet Protocol network connection

PowerHA relies on a Internet Protocol network connection to communicate between nodes. Before you configure PowerHA, check that the network is configured correctly. To check network configurations for the cluster, complete the following steps:

1. Check that the IP addresses are configured in the network interfaces by running the following command:

```
#netstat -in | grep -v link
```

Example 7-57 shows the configuration of the IP addresses on both nodes.

Example 7-57 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 that the /etc/hosts file has all the entries of the IP addresses and that their labels are the same ones that are used by PowerHA:

```
#vi /etc/hosts
```

Example 7-58 shows an example of a hosts file.

Example 7-58 IP entries in hosts file

```
9.43.86.111      service      ## Service IP address

192.168.10.101  node1_boot1   ## PowerHA N/W interface on node1
192.168.20.101  node1_boot2   ## PowerHA N/W interface on node1
192.168.30.101  node1_hadr    ## HADR N/W interface on node1

192.168.10.102  node2_boot1   ## PowerHA N/W interface on node2
192.168.20.102  node2_boot2   ## PowerHA N/W interface on node2
192.168.30.102  node2_hadr    ## HADR N/W interface on node2
```

3. Verify that name resolution is working by running **host**. If something is wrong, check and modify the /etc/hosts file.

```
#host node1_boot1
node1_boot1 is 192.168.10.101.
```

4. Check the serial network connection:

- a. Check that both machines are connected by an RS232 cable.
- b. Configure the tty device on machine #1 and machine #2 by running the following command:

```
# smitty tty
```

Select Add a TTY → tty rs232 Asynchronous Terminal → sa0 Available 01-S1 Standard I/O Serial Port. In the Add a TTY panel (Figure 7-12), press F4 to show the list of available port numbers. Select the port number that is 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	[dtopen]	+
[MORE...30]		

Figure 7-12 Add a TTY

- c. Check that the tty device is available by running the following command:

```
# lsdev -Cc tty
tty0 Available 01-S1-00-00 Asynchronous Terminal
```

Configuring PowerHA

The AIX **smitty** interface provides PowerHA configuration interfaces. The main steps to configure PowerHA are as follows:

1. Add nodes to an PowerHA cluster.
2. Add a service IP label/address.
3. Configure the application servers.
4. Add a resource group.
5. Add resources to the resource group.
6. Configure the persistent IP alias for HADR communication.
7. Define the serial network and the serial network device.
8. Verify and synchronize PowerHA configurations.
9. Conduct a basic verification of the PowerHA configuration.

Here are these steps in detail:

1. Add nodes to an PowerHA cluster by running the following command:

```
#smitty sysmirror
```

You can run **smitty hacmp** instead.

From the PowerHA SystemMirror menu, select Cluster Nodes and Networks → Manage Nodes → Add Nodes to an PowerHA Cluster.

In Configure Nodes to an PowerHA Cluster (standard) menu (Figure 7-13), enter the cluster name and new nodes.

Add a Node	
Type or select values in entry fields.	
Press Enter AFTER making all desired changes.	
[Entry Fields]	
* Node Name	[node1_boot1]
Communication Path to Node	[] +

Figure 7-13 Add node to PowerHA cluster

2. Add a service address to the resource group by running the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resources → 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 (Figure 7-14).

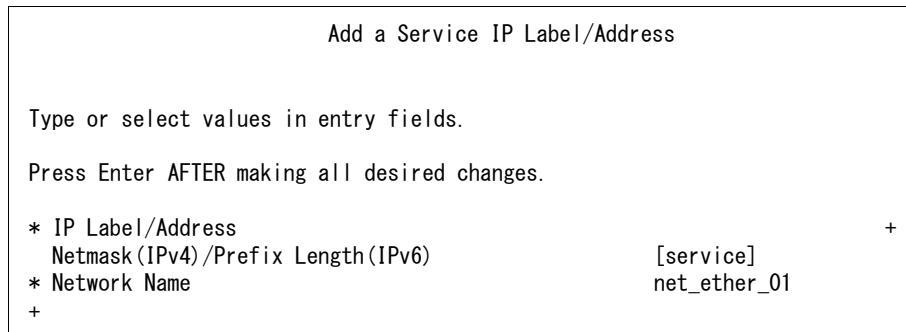


Figure 7-14 Add a service IP label

3. Prepare the application server start/stop scripts.

In this example, we use dummy start and stop scripts to test the PowerHA setup. We provide the full functional start and stop scripts to control the HADR database in a PowerHA environment in 7.5.5, “Preparing the application server scripts” on page 326.

Example 7-59 is a dummy start/stop script that records the time when the scripts are run. We place the scripts in the /home/hadrinst/scripts directory. Copy this dummy script and change its name as follows:

- Start script of PowerHA application server:

/home/hadrinst/scripts/hadr_primary_takeover.ksh

- Stop script of PowerHA application

server:/home/hadrinst/scripts/hadr_primary_stop.ksh

Example 7-59 Dummy application server start/stop script

```
#!/usr/bin/ksh -x

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####
date
echo "#####

exit 0
```

Ensure that these scripts have execution permissions by running the following command:

```
#ls -l /home/hadrinst/scripts
```

4. Configure the application server.

To access this menu, run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resources → Configure User Applications (Scripts and Monitors) → Application Controller Scripts → Add Application Controller Scripts.

In the Add Application Controller Scripts menu, enter the application controller name, and complete the paths of the start and stop scripts (Figure 7-15).

Add Application Controller Scripts	
Type or select values in entry fields.	
Press Enter AFTER making all desired changes.	
* Application Controller Name	[hadr_server]
* Start Script	[/home/hadrinst/scripts/hadr_primary_takeover.ksh]
* Stop Script	[/home/hadrinst/scripts/hadr_primary_stop.ksh]
Application Monitor Name(s)	

Figure 7-15 Adding an application controller

5. Add a resource group.

To add a resource group, run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resource Groups → Add a Resource Group.

In the Add a Resource Group menu, enter the Resource group name and the participating node names (Figure 7-16). This information corresponds to a *Rotating resource group* in the earlier PowerHA versions, which means that there is no priority for a resource group between nodes. For more information, see *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Planning PowerHA SystemMirror*, found at:

http://pic.dhe.ibm.com/infocenter/aix/v6r1/topic/com.ibm.aix.powerha.plangd/hacmpplangd_pdf.pdf

The screenshot shows a terminal window titled 'Add a Resource Group'. It contains the following text:

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 7-16 Adding a resource group

6. Add resources to the resource group by running the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Applications and Resources → Resource Groups → Change>Show Resources and Attributes for a Resource Group.

In the Change/Show Resources for a Cascading Resource Group menu, enter the service IP label/address, service, as shown in Figure 7-17.

Change/Show All Resources and Attributes for a Custom Resource Group		
Type or select values in entry fields.		
Press Enter AFTER making all desired changes.		
[Entry Fields]		
Resource Group Name	hadr_rg	
Participating Nodes (Default Node Priority)	node1 node2	
Startup Policy	Online Using Distribution Policy	
Failover Policy	Failover To Next Priority Node In The List	
Fallback Policy	Never Fallback	
Service IP Labels/Addresses	[service]	+
Application Controllers	[hadr_server]	+
Volume Groups	[]	+
Use forced varyon of volume groups, if necessary	false	+
Filesystems (empty is ALL for VGs specified)	[]	+

Figure 7-17 Adding resource to a resource group

7. Define the subsidiary network and serial network device.

A subsidiary network for PowerHA keep-alive makes PowerHA failure detection more secure. For more details, see *PowerHA SystemMirror Version 7.1 for AIX Standard Edition: Administering PowerHA*, found at:

http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.powerha.admngd/hacmpadmngd_pdf.pdf

Run the following command:

```
# smitty sysmirror
```

From the PowerHA SystemMirror menu, select Cluster Nodes and Networks → Discover Networks Interfaces and Disks.

In the list of Discovered Communication Devices, choose the tty device on each node you connected with an RS-232C cable. In our example, we select node1 tty2 and node2 tty2.

8. Verify and synchronize PowerHA configurations.

After you define the PowerHA configuration on one node, you must verify and synchronize the cluster topology on the other node. Run the following command:

```
# smitty sysmirror
```

From the PowerHA for SystemMirror menu, select Cluster Nodes and Networks → Verify and Synchronize Cluster Configuration.

The PowerHA verification utility checks that the cluster definitions are the same on all the nodes and provides diagnostic messages if errors are found.

Starting PowerHA

The PowerHA configuration is now complete. Now start the PowerHA cluster by completing the following steps:

1. Start PowerHA on the service node Node1.

To start PowerHA from the **smit** menu, run the following command:

```
# smitty clstart
```

In the Start Cluster Services menu, select “now” for “*Start now, on system restart or both”. Select **true** for “Startup Cluster Information Daemon?” and leave the default for all the other fields (Figure 7-18).

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]	+
Manage Resource Group	false	+
BROADCAST message at startup?	false	+
Startup Cluster Information Daemon?	true	+
+		

Figure 7-18 Starting cluster services

Check the `cluster.log` file to see if PowerHA started successfully by running the following command:

```
# tail -f /usr/es/adm/cluster.log
```

The message `EVENT COMPLETED: node_up_complete xxx` indicates that PowerHA started successfully, where `xxx` is the node number. Figure 7-19 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 7-19 Cluster.log output

2. Start PowerHA on standby node Node2.

After the PowerHA service node is started, you can use the same procedure to start the standby node.

After PowerHA is started on both nodes, check the following items:

- ▶ Check the status of the resource group by the PowerHA command `c1RGinfo`. You can run this command on either node. Example 7-60 shows the out put of this command. You can see that resource group `hadr_rg` is `OFFLINE` on `node1` and `ONLINE` on `node2`, which means that the resource group `hadr_rg` is now acquired by `node2`.

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
```

Example 7-60 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 by running the following command:

```
#netstat -i | grep -v link
```

Example 7-61 shows the output of the **netstat** command. The service IP should be in the output list.

Example 7-61 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, ensure that the application server is started.

When the PowerHA 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 7-62 shows the example output. Because now we set a dummy script, nothing happens on HADR databases. The script records only the time when the script is issued to the log file.

Example 7-62 Check application server

```
hadr_rg:[6] echo
#####
#####hadr_rg:[7] date
Tue Jun 17 20:04:10 CDT 2012
```

PowerHA takeover test

You can test the PowerHA takeover function by stopping PowerHA on the service node in takeover mode. The standby node should take over the resource group after the takeover operation. To stop PowerHA in the takeover mode, run **smitty** in service node (Node1 in our example):

```
#smitty clstop
```

In the Stop Cluster Services menu, select “now” for “Stop now, on system restart or both”. Select the node where PowerHA is to be stopped, and select “takeover” for “Shutdown mode” (Figure 7-21 on page 325).

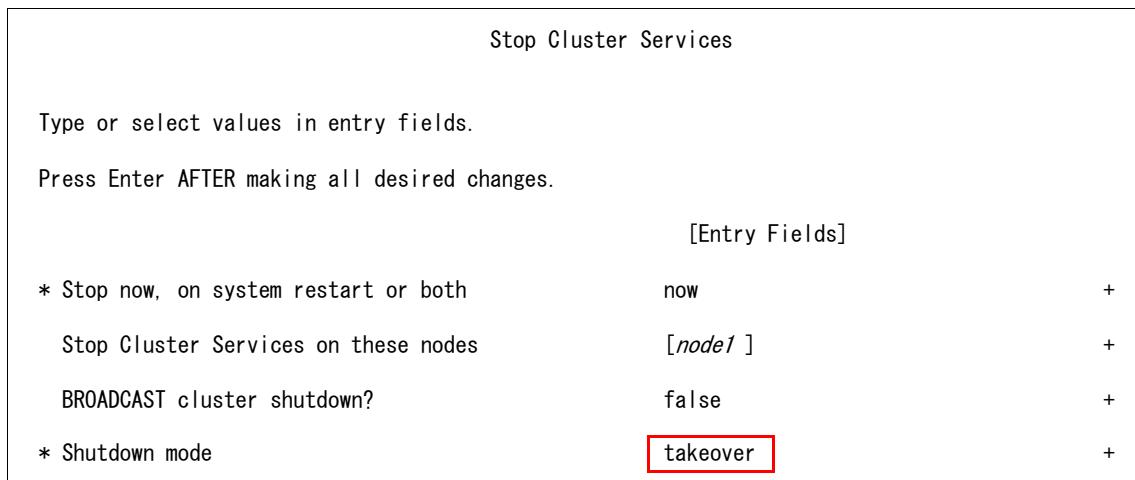


Figure 7-20 Stop PowerHA with takeover

- ▶ Check whether the PowerHA takeover process stopped the service node. To check if this task is complete, run the following command:

```
# tail -f /var/hacmp/adm/cluster.log
```

The PowerHA 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 (Example 7-63).

Example 7-63 Check takeover result

```
# tail -f /var/hacmp/adm/cluster.log
.....
Aug 05 20:59:55 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 stopped successfully by running the following command:

```
# cat /tmp/hadr_primary_stop.ksh.log
```
- ▶ On the service node Node1, check if the service address is released by running the following command:

```
# netstat -i
```
- ▶ On standby node Node2, check that the node_down_complete Node1 event is completed by running the following command:

```
# tail -f /usr/es/adm/cluster.log
```

Example 7-64 shows a snippet of the cluster.log file.

Example 7-64 Check node_down_complete event from Node 2

```
# tail -f /var/hacmp/adm/cluster.log
.....
Aug 05 20:59:55 AIX: EVENT COMPLETED: node_down_complete node1
.....
```

- ▶ Check that Node2 owns the resource group by running the **cLRGinfo** command (Example 7-65).

Example 7-65 Checking if Node2 took over the resource

```
# /usr/es/sbin/cluster/utilities/cLRGinfo
```

Group	Name	State	Node
<hr/>			
hadr_rg		OFFLINE	node1
		ONLINE	node2

- ▶ Check that Node2 took over the service IP address by running **netstat** in Node2 (Example 7-66).

Example 7-66 Check service IP address

```
root@node2:/home/hadrinst/scripts# netstat -i | grep -v link
Name  Mtu   Network      Address          Ipkts  Ierrs    Opkts
Oerrs  Coll
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 dummies, HADR takeover is not issued yet. Here we check if PowerHA functions work correctly by running the following command:

```
# cat /tmp/hadr_primary_takeover.ksh.log
```

Stopping PowerHA

To stop PowerHA, run **smitty** on both Node1 and Node 2:

```
#smitty c1stop
```

In the Stop Cluster Services menu, select “now” for “Stop now, on system restart or both”. Select the node to be stopped and select “graceful” for “Shutdown mode” (Figure 7-21).

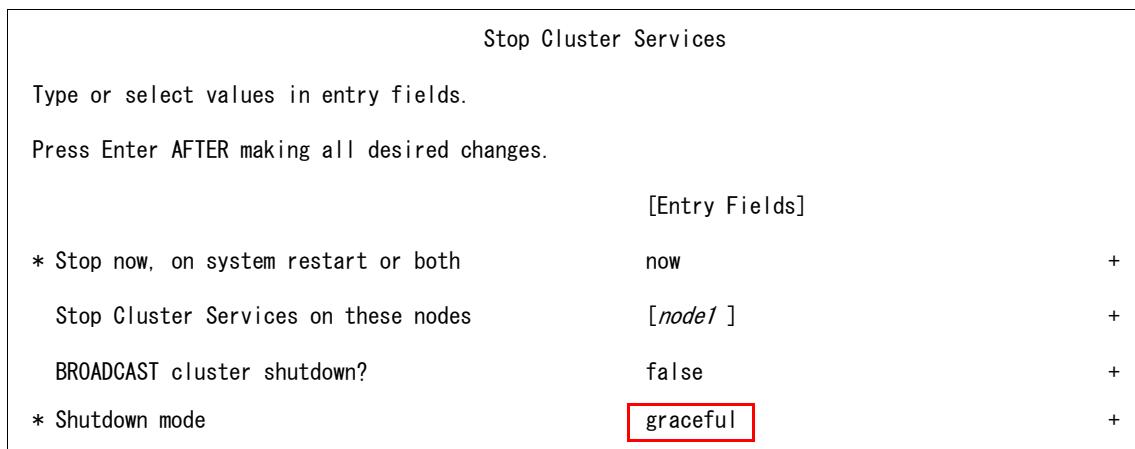


Figure 7-21 Stop PowerHA

7.5.5 Preparing the application server scripts

To automate HADR takeover with PowerHA, use the PowerHA application server start/stop scripts. By including the HADR commands in the start/stop scripts, PowerHA can handle HADR operations 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, “PowerHA application server scripts” on page 533.

Here we explain some of these PowerHA 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 the PowerHA application server. This script is run in the following situations when the node acquires the resource group:

- ▶ Normal start of PowerHA on the first (service) node in the cluster.
- ▶ Unplanned takeover that is triggered by failure detection of PowerHA.
- ▶ Planned takeover by stopping PowerHA in takeover mode, or moving the resource group by running an administrative command.

This script is run not only when the resource group fails 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 running the PowerHA command `c1RGinfo`.

Figure 7-22 shows the flow chart of the script, `hadr_primary_takeover.ksh`. If the trigger initiates a normal PowerHA start, then this script simply exits. If the starting trigger is takeover, which means that this script is run on the standby node with the resource group fallover, then this script checks the HADR role. If the HADR role is standby, it runs the `takeover hadr` command with the `by force` option.

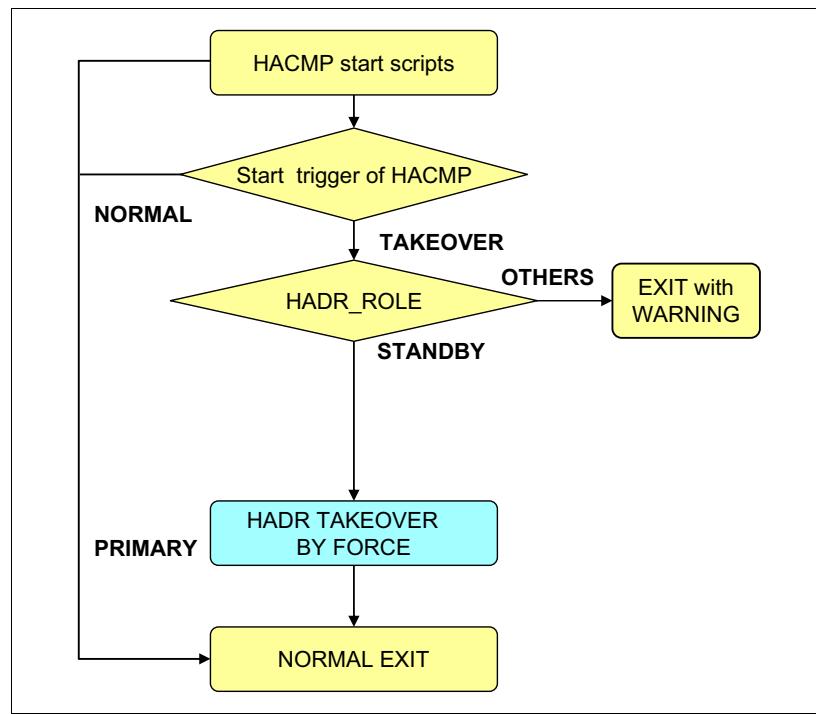


Figure 7-22 `hadr_primary_takeover.ksh` script flow

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, the `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.

hadr_primary_stop.ksh (stop script)

We define **hadr_primary_stop.ksh** as the stop script for the PowerHA application server. Figure 7-23 shows the script logic flow. This script is run on the service node and when the node releases the resource group, that is, when you stop PowerHA 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 run the **takeover hadr** command as soon as possible. In our example, this script runs a remote command to the standby node to run the **takeover hadr** command.

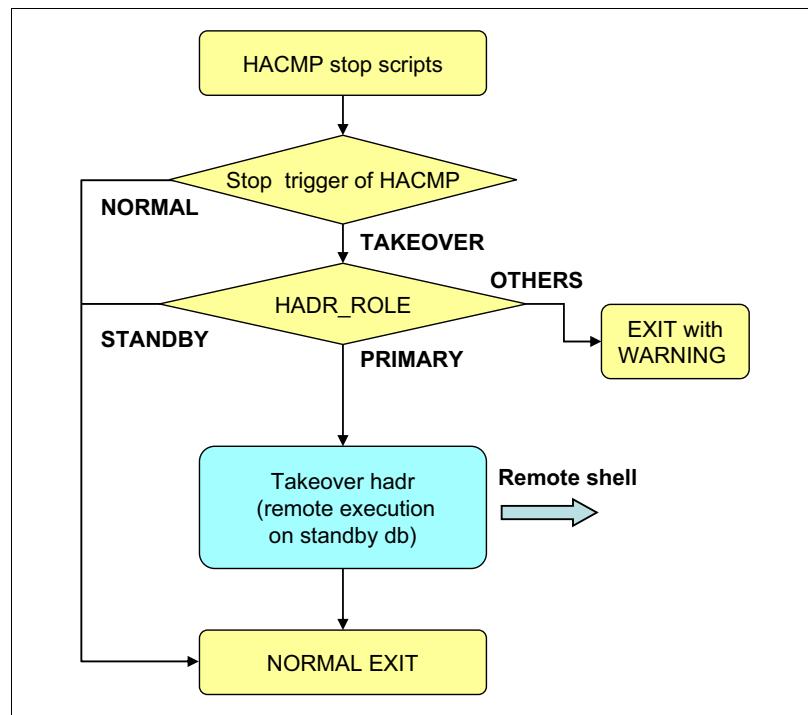


Figure 7-23 *hadr_primary_stop.ksh* script flow

Additional information for PowerHA commands

You can use the output of the PowerHA 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 a normal start (Example 7-67), and two nodes when the start trigger is a takeover (Example 7-68).

Example 7-67 Normal start on node1

Group Name	Type	Resource Group Movement
hadr_rg	non-concurrent	PRIMARY=" :node1"

Example 7-68 Takeover from node1 to node2

Group Name	Type	Resource Group Movement
hadr_rg	non-concurrent	PRIMARY="node1:node2"

7.5.6 Joint test for HADR and PowerHA

After the functional scripts replace the dummy scripts, you can test the combination of HADR automatic takeover in an PowerHA environment. We implement the joint test for the following scenarios:

► Planned failover:

In the first scenario, we use the facility that is provided by the PowerHA 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 HADR and PowerHA in the cluster

To carry out the test scenarios, complete the following steps to start HADR and PowerHA:

1. Start the standby database on node2.

Start the HADR databases that you set up in 7.5.3, “HADR setup” on page 311. Start the standby database on Node2 first:

- Check that HADR role is standby on node2 by running the following command:
\$db2 get db cfg for sample
- Start the standby database by running the following command:
\$db2 activate db sample

Tip: When you start the HADR process, you do not have to run **start hadr** or **stop hadr** every time. After you start an HADR database in a primary or standby role, an activate and deactivate database command 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 by running the following command:
\$db2 get db cfg for sample
- Start the standby database by running the following command:
\$db2 activate db sample
- Check if the HADR status is in the Peer state by running the following command:
\$db2pd -hadr -db sample

3. Start PowerHA on Node1. For details, see “Starting PowerHA” on page 320.

4. Start PowerHA on Node2. For details, see “Starting PowerHA” on page 320.

5. Check the HADR and PowerHA status to see if they are ready for the test:

- Check if HADR is in the Peer state by running the following command:
\$db2pd -hadr -db sample
- Check if the resource group is ONLINE on node1 by running the following command:
/usr/es/sbin/cluster/utilities/c1RGinfo

Configuring the client node

To check the client access after takeover, configure the client machine by completing the following steps:

1. Create an instance on the client node by running the following command:

```
$db2icrt -u clntinst clntinst
```

2. Catalog a node directory by specifying the service address service by running the following command:

```
$db2 catalog tcpip node SERVICE remote service server 50030
```

3. Catalog the database by running the following command:

```
$ db2 catalog database sample at node SERVICE
```

4. Connect to database server by running the following command:

```
$ db2 connect to sample user hadrinst using hadrinst
```

5. Run the query that is shown in Example 7-69 to check which server you are connecting to. You can see that we are now connected to Node1.

We configured the client machine on Node3.

Example 7-69 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 that is provided by the PowerHA application to test the failover between the two nodes. Complete the following steps:

1. Stop PowerHA in takeover mode on service node Node1 by running the following command:

```
#smitty clstop
```

In Stop Cluster Services menu, select “now” on “Stop now, on system restart or both”. Select the node where PowerHA is to be stopped. Select “takeover” on “Shutdown mode”.

2. Check if the PowerHA takeover process stopped the service node. You can accomplish this task by running the following command:

```
# tail -f /usr/es/adm/cluster.log
```

The PowerHA 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.

3. Check that the **stop** script is run on Node1 and that **TAKEOVER HADR** is issued remotely on Node2 by running the following command:

```
#cat hadr_primary_stop.ksh.log
```

You see the output in Example 7-70 in the log file of this script.

Example 7-70 Output of stop script

```
....  
HADR takeover is issued on remote node  
....  
node2: DB20000I The TAKEOVER HADR ON DATABASE command completed  
successfully.
```

4. Check the status of PowerHA on Node2. The resource group should be taken over to Node2 (Example 7-71).

Example 7-71 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 nodes by running the following command:

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

Now the primary database is running on Node2 and the standby database is running on Node1.

6. Reconnect from the client node Node3, and run the query that is shown in Example 7-72. You can see that you 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 ACR.

Example 7-72 Check the 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 PowerHA 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 fall back the resource group from Node2 to Node1, you can stop PowerHA in takeover mode on Node2.

9. Start PowerHA on Node2.

The primary database is running on Node1 and the standby database is running on Node2. Now the cluster returns to the state we started with.

Unplanned failover for a system crash

In this scenario, we test failover by halting the system intentionally. We check that PowerHA detects the failure and that the standby database is switched to the primary database with the PowerHA resource group takeover.

Complete the following steps:

1. Check the PowerHA and HADR status of the two nodes:
 - PowerHA service is started on both nodes.
 - HADR is in the Peer state.
2. Connect from the client and issue some queries by completing the steps in “Configuring the client node” on page 331.
3. On Node1, run the following command as root:

```
#sync;sync;sync;halt -q
```

Node1 terminates processing immediately. PowerHA detects the outage and starts failover of the resource group to Node2.

4. Check that the resource group is taken over by Node2 by running the following command:

```
# /usr/es/sbin/cluster/utilities/cLRGinfo
```

5. Check that the start script is issued on Node2 by running the following command:

```
# view /tmp/hadr_primary_takeover.ksh.log
```

6. Check that the primary HADR database is now on Node2 by running the following command:

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

If the HADR primary database is on Node2, the failover is successful.

Next, we want to fail back the resource group to the original node, as the failed system situation is repaired. Power on Node1 and start DB2 instance by running the following command:

```
$db2start
```

You must reintegrate the old primary database to the current state by using the HADR function. In the old primary system (Node1), complete the following steps:

1. Check the HADR role on Node1. The HADR role on Node1 should still show as primary (Example 7-73).

Example 7-73 Check HADR role

```
$ db2 get db cfg for sample | grep "HADR database role"
```

HADR database role	= PRIMARY
--------------------	-----------

2. Check the **db2diag.log** file by running one of the following commands to see the HADR catching up process. After you start the HADR database as standby on Node1, the catching up process starts.

- \$ db2diag -f

- \$ tail -f /home/hadrinst/sql1lib/db2dump/db2diag.log

3. Restart the HADR database as standby on Node1.

To reintegrate the database on Node1 into the HADR pair as a standby database, complete the following steps:

- a. To switch the role of the database from primary to standby, you must run **start hadr** instead of **activate database** (Example 7-74). Check the messages in the db2diag.log until they return to the peer status.

Example 7-74 Start database as HADR standby

```
$db2 "start hadr on db sample as standby"
DB20000I The START HADR ON DATABASE command completed
successfully.
```

- b. Check if the HADR status came back to the Peer state after it catches up the log from the primary on Node1 as follows:
 - The **db2pd** command shows the current HADR status.
 - In the db2diag.log file, you see the messages that are shown in Example 7-75.

Example 7-75 db2diag.log message

```
2012-08-05-22.06.03.317330-240 E23808E369           LEVEL: Event
PID      : 9634             TID   : 47103946516800PROC :
db2sysc
INSTANCE: hadrinst1          NODE  : 000
EDUID    : 27               EDUNAME: db2hadrp (SAMPLE)
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to S-Peer (was S-NearlyPeer)
```

4. Start PowerHA on Node1 again. Check that there is no change in the HADR status and that the resource group is still ONLINE on Node2.
5. Fall back from Node2 to Node1. To fall back the resource group from Node2 to Node1, you can stop the PowerHA in takeover mode on Node2.
6. Start PowerHA on Node2.

After the primary database is running on Node1 and the standby database is running on Node2, you can start PowerHA on Node2. The cluster now returns to the state we started from.

Stopping PowerHA and HADR

Complete the following steps to stop the PowerHA system for a maintenance operation:

1. Stop PowerHA on the Service node and the standby node by following the steps in “Stopping PowerHA” on page 325.
2. Stop HADR.
 - Stop HADR on the primary database on the service node by running the following command:
\$db2 deactivate db sample
 - Stop the instance on the service node by running the following command:
\$db2stop
 - Stop HADR on the standby database on the standby node by running the following command:
\$db2 deactivate db sample
 - Stop the instance on the standby node by running the following command:
\$db2stop

Stopping the HADR process: The **deactivate database** command can be used to stop the HADR process instead of the **stop hadr** command. When you run **deactivate database**, the HADR database role stays in the database configuration. If you run **stop hadr**, the HADR database role changes to STANDARD, which makes it hard for you to know which node was primary and which node was standby.



HADR monitoring

Monitoring is a part of setting up and maintaining a high availability disaster recovery (HADR) setup. The DB2 monitoring interfaces provide a detailed picture of the configuration and health of a HADR environment.

This chapter describes two methods of acquiring the status of a HADR environment: the **db2pd** command and the **MON_GET_HADR** table function.

This chapter covers the following topics:

- ▶ Introduction to HADR monitoring
- ▶ The db2pd command
- ▶ The MON_GET_HADR table function
- ▶ HADR monitoring information

8.1 Introduction to HADR monitoring

Starting with DB2 10.1, there are two ways to monitor a HADR environment:

- ▶ The **db2pd** command.
- ▶ The **MON_GET_HADR** table function.

In DB2 10.1, the **MON_GET_HADR** table function is introduced to report information about HADR functionality. Also, the **db2pd** command is changed to report new HADR information. The HADR information reported by these interfaces includes details about new functionality, such as the HADR multiple standby mode.

The information that is returned by these interfaces depends on where they are issued. For example, monitoring on a standby database returns information about that standby and the primary database only; no information is provided about any other standbys. Monitoring on the primary database returns information about all of the standbys if you are using the **db2pd** command or the **MON_GET_HADR** table function. Even standbys that are currently not connected, but are configured in the primary's **hadr_target_list** configuration parameter, are displayed.

The **db2pd** command and the **MON_GET_HADR** table function return essentially the same information, but the **db2pd** command does not require reads on standby to be enabled (for reporting from a standby). The **db2pd** command is preferred during takeover because there could be a time window where the primary or the standby does not allow client connections.

The following methods can also be used to monitor a HADR environment, but starting with DB2 10.1, they are deprecated, and they might be removed in a future release:

- ▶ The **GET SNAPSHOT FOR DATABASE** command: This command collects status information and formats the output. The information that is returned is a snapshot of the database manager operational status at the time that you run the command. HADR information is displayed in the output under the heading HADR status.
- ▶ The **db2GetSnapshot** API: This API collects database manager monitor information and writes it to a user-allocated data buffer. The information that is returned is a snapshot of the database manager operational status when the API was called.
- ▶ The **SNAPHADR** administrative view and the **SNAP_GET_HADR** table function: This administrative view and this table function return information about HADR from a database snapshot, in particular, the HADR logical data group.

- ▶ Other snapshot administrative views and table functions: You can use the following snapshot administrative views and table functions, which are not HADR specific and return a wider set of information, to query a subsection of the HADR information:
 - **ADMIN_GET_STORAGE_PATHS**
 - **MON_GET_TRANSACTION_LOG**
 - **SNAPDB**
 - **SNAPDB_MEMORY_POOL**
 - **SNAPDETAILLOG**
 - **SNAP_GET_DB**
 - **SNAP_GET_DB_MEMORY_POOL**

You can still use all these deprecated methods, but the HADR information that is reported does not include all of the details about new functionality. For example, if you call the **SNAPHADR** administrative view or **SNAP_GET_HADR** table function from a primary database, they do not report information about any auxiliary standby databases.

In the following subsections, we describe the **db2pd** command and the **MON_GET_HADR** table function in detail.

For more information about deprecated methods for monitoring a HADR environment, see:

- ▶ The **GET SNAPSHOT FOR DATABASE** command:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.1uw.admin.cmd.doc%2Fdoc%2Fr0001945.html>
- ▶ The **db2GetSnapshot** API:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.1uw.apdv.api.doc%2Fdoc%2Fr0001449.html>
- ▶ The **SNAPHADR** administrative view and the **SNAP_GET_HADR** table function:
<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.1uw.sql.rtn.doc%2Fdoc%2Fr0021981.html>
- ▶ Other snapshot administrative views and table functions:
http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/nav/3_6_1_3_12

8.2 The db2pd command

The **db2pd** command retrieves information from the DB2 memory sets. In a HADR environment, it can be run from either a primary or a standby database. If there are multiple standby databases and this command is run from a standby, it does not return any information about the other standbys. If this command is issued from the primary, it returns information about all standbys.

To view information about primary and standby databases in a HADR environment, run the following command:

```
db2pd -db <databasename> -hadr
```

In the following example, the **db2pd -hadr** command is run in a HADR environment on a primary database called “Sample” (node1) with three standbys (node2, node3, and node4). Example 8-1 shows the resulting output. Three sets of data are returned, with each one representing a primary-standby log shipping channel.

Example 8-1 Sample db2pd output for a HADR setup with three standby databases

```
db2pd -db Sample -hadr
```

```
Database Member 0 -- Database Sample -- Active -- Up 0 days 00:23:17 --
Date 10/07/2012 13:57:23
```

```
HADR_ROLE = PRIMARY
REPLAY_TYPE = PHYSICAL
HADR_SYNCMODE = SYNC
STANDBY_ID = 1
LOG_STREAM_ID = 0
HADR_STATE = PEER
PRIMARY_MEMBER_HOST = node1.ibm.com
PRIMARY_INSTANCE = db2inst1
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = node2.ibm.com
STANDBY_INSTANCE = db2inst2
STANDBY_MEMBER = 0
HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 10/07/2012 13:38:10.199479
HEARTBEAT_INTERVAL(seconds) = 30
HADR_TIMEOUT(seconds) = 120
TIME_SINCE_LAST_RECV(seconds) = 3
PEER_WAIT_LIMIT(seconds) = 0
LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
```

```

LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
    LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 50772
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87616
    PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
    STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
    HADR_LOG_GAP(bytes) = 0
STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
    STANDBY_RECV_REPLY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 10/07/2012 13:49:19.000000
        STANDBY_LOG_TIME = 10/07/2012 13:49:19.000000
    STANDBY_REPLY_LOG_TIME = 10/07/2012 13:49:19.000000
STANDBY_RECV_BUF_SIZE(pages) = 16
    STANDBY_RECV_BUF_PERCENT = 0
    STANDBY_SPOOL_LIMIT(pages) = 0
        PEER_WINDOW(seconds) = 0
    READS_ON_STANDBY_ENABLED = Y
STANDBY_REPLY_ONLY_WINDOW_ACTIVE = N

    HADR_ROLE = PRIMARY
    REPLAY_TYPE = PHYSICAL
    HADR_SYNCMODE = SUPERASYNC
    STANDBY_ID = 2
    LOG_STREAM_ID = 0
        HADR_STATE = REMOTE_CATCHUP
PRIMARY_MEMBER_HOST = node1.ibm.com
    PRIMARY_INSTANCE = db2inst1
    PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = node3.ibm.com
    STANDBY_INSTANCE = db2inst3
    STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
    HADR_CONNECT_STATUS_TIME = 10/07/2012 13:35:51.724447
    HEARTBEAT_INTERVAL(seconds) = 30
        HADR_TIMEOUT(seconds) = 120
TIME_SINCE_LAST_RECV(seconds) = 16
    PEER_WAIT_LIMIT(seconds) = 0
    LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
    LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
    PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
    STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315

```

```

        HADR_LOG_GAP(bytes) = 0
STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_RECV_RECV_GAP(bytes) = 0
            PRIMARY_LOG_TIME = 10/07/2012 13:49:19.000000
            STANDBY_LOG_TIME = 10/07/2012 13:49:19.000000
        STANDBY_RECV_LOG_TIME = 10/07/2012 13:49:19.000000
STANDBY_RECV_BUF_SIZE(pages) = 16
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 0
            PEER_WINDOW(seconds) = 0
        READS_ON_STANDBY_ENABLED = Y

        HADR_ROLE = PRIMARY
        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = SUPERASYNC
        STANDBY_ID = 3
        LOG_STREAM_ID = 0
        HADR_STATE = REMOTE_CATCHUP
PRIMARY_MEMBER_HOST = node1.ibm.com
        PRIMARY_INSTANCE = db2inst1
        PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = node4.ibm.com
        STANDBY_INSTANCE = db2inst4
        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 10/07/2012 13:46:51.561873
        HEARTBEAT_INTERVAL(seconds) = 30
            HADR_TIMEOUT(seconds) = 120
TIME_SINCE_LAST_RECV(seconds) = 6
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.006298
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.516
        LOG_HADR_WAIT_COUNT = 82
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
            PRIMARY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
            STANDBY_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
                HADR_LOG_GAP(bytes) = 0
STANDBY_RECV_RECV_LOG_FILE,PAGE,POS = S0000009.LOG, 1, 49262315
        STANDBY_RECV_RECV_GAP(bytes) = 0
            PRIMARY_LOG_TIME = 10/07/2012 13:49:19.000000
            STANDBY_LOG_TIME = 10/07/2012 13:49:19.000000
        STANDBY_RECV_RECV_LOG_TIME = 10/07/2012 13:49:19.000000
STANDBY_RECV_BUF_SIZE(pages) = 16

```

```
STANDBY_RECV_BUF_PERCENT = 0
STANDBY_SPOOL_LIMIT(pages) = 0
PEER_WINDOW(seconds) = 0
READS_ON_STANDBY_ENABLED = N
```

The HADR_ROLE field represents the role of the database on which **db2pd** is run, so it is listed as PRIMARY in all sets. The HADR_STATE for the two auxiliary standbys (node3 and node4) is REMOTE_CATCHUP because they automatically run in SUPERASYNC mode (which is also reflected in the **db2pd** output). The STANDBY_ID differentiates the standbys. It is system generated and the ID-to-standby mapping can change from query to query; however, the ID “1” is always assigned to the principal standby.

Fields that are not relevant to the current settings are omitted in the output. For example, in Example 8-1 on page 340, information about the replay-only window (like start time and transaction count) is not included because reads on standby is not enabled.

Section 8.4, “HADR monitoring information” on page 345 describes all HADR monitoring information that is returned by the **db2pd -hadr** command.

For more information about the **db2pd -hadr** command, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.cmd.doc%2Fdoc%2Fr0011729.html>

8.3 The MON_GET_HADR table function

As an alternative to the **db2pd** command, you can use the **MON_GET_HADR** table function to monitor a HADR environment. Both commands essentially return the same information.

With the **MON_GET_HADR** table function, you can query for particular HADR monitoring information for a specific database member of a HADR environment. For the input argument of the **MON_GET_HADR** table function, you provide the number of the database member you want to monitor. Specify -1 for the current database member, or -2 for all database members. If the null value is specified, -1 is set implicitly. As a result, the **MON_GET_HADR** table function provides monitoring information for the specified member according to the parameter list defined in the select clause of the query from which the **MON_GET_HADR** table function is run.

If you run this query on the primary database, it return information about all the standbys. If you want to run the **MON_GET_HADR** function against a standby database, be aware of the following points:

- ▶ You must enable reads on standby.
- ▶ Even if your HADR setup is in multiple standby mode, the table function does not return any information about any other standbys.

In Example 8-2, a DBA calls the **MON_GET_HADR** table function on a primary database (node1) with three standbys (node2, node3, and node4). Three rows are returned. Each row represents a primary-standby log shipping channel. The **HADR_ROLE** column represents the role of the database to which the query is issued. Therefore, it is **PRIMARY** on all rows. The **HADR_STATE** for the two auxiliary standbys (node3 and node4) is **REMOTE_CATCHUP** because they automatically run in **SUPERASYNC** mode.

Example 8-2 Sample using the MON_GET_HADR table function

```
db2 "select HADR_ROLE, STANDBY_ID, HADR_STATE,
      varchar(PRIMARY_MEMBER_HOST,20)
      as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST,20) as
      STANDBY_MEMBER_HOST from table (mon_get_hadr(NULL))"

HADR_ROLE STANDBY_ID HADR_STATE      PRIMARY_MEMBER_HOST
STANDBY_MEMBER_HOST
-----
-----
PRIMARY   1          PEER           node1.ibm.com      node2.ibm.com
PRIMARY   2          REMOTE_CATCHUP  node1.ibm.com      node3.ibm.com
PRIMARY   3          REMOTE_CATCHUP  node1.ibm.com      node4.ibm.com

3 record(s) selected.
```

Section 8.4, “HADR monitoring information” on page 345 describes in detail all the information that is returned by the **MON_GET_HADR** table function.

For more information about the **MON_GET_HADR** table function, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.sql.rtn.doc%2Fdoc%2Fr0059275.html>

8.4 HADR monitoring information

This section describes the HADR monitoring information that is provided by the `db2pd -hadr` command and the `MON_GET_HADR` table function. In particular, all the following elements can be reported:

- ▶ **HADR_ROLE:**

The current HADR role of the local database. Possible values are:

- PRIMARY
- STANDBY

- ▶ **REPLAY_TYPE**

The type of HADR replication of the database. The possible value is PHYSICAL.

- ▶ **HADR_SYNCMODE**

The current HADR synchronization mode of the local database. Possible values are:

- ASYNC

In this mode, log writes are considered successful only when the log records are written to the log files on the primary database and are 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 while they are still on their way to the standby database.

- NEARSYNC

In this mode, log writes are considered successful only when the log records are written to the log files on the primary database and when the primary database receives an acknowledgement from the standby system that the logs are written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site does not transfer all of the log data that it received to nonvolatile storage.

- SUPERASYNC

In this mode, the HADR pair can never be in a Peer state or disconnected Peer state. The log writes are considered successful when the log records are written to the log files on the primary database. Because the primary database does not wait for acknowledgement from the standby database, transactions are considered committed regardless of the state of the replication of that transaction.

- SYNC: In this mode, log writes are considered successful only when logs are written to log files on the primary database and when the primary database receives an acknowledgement from the standby database that the logs are written to log files on the standby database. The log data is guaranteed to be stored at both sites.

In multiple standby mode, the HADR_SYNCMODE value of the standby is shown as an empty string (a zero-length string) until the primary connects to the standby database.

- ▶ STANDBY_ID

The identifier for all the standbys in the current setup. This value has meaning only when the command is run on the primary. If you run it on a standby, it always returns 0 because, even in multiple standby mode, other standbys are not visible to each other. The 1 identifier is always assigned to the standby in single standby mode, while in multiple standby mode, 1 indicates the principal standby.

- ▶ LOG_STREAM_ID

The identifier for the log stream that is being shipped from the primary database.

- ▶ HADR_STATE

The current HADR state of the standby database. Possible values are:

- DISCONNECTED

The primary and standby databases are disconnected.

- DISCONNECTED_PEER

If you configure a peer window and the primary database loses its connection with the standby database in the Peer state, the primary database continues to behave as though the primary and standby databases are in the Peer state for the configured amount of time (called the peer window), or until the standby reconnects, whichever happens first. When the primary database and standby database are disconnected but behave as though they are in the Peer state, this state is called disconnected peer.

- LOCAL_CATCHUP

With the HADR feature, when a standby database is started, it enters a local catchup state and the log files in its local log path are read to determine what logs are available locally. In this state, logs are not retrieved from the archive even if you configure a log archiving method.

Also, in this state, a connection to the primary database is not required; however, if a connection does not exist, the standby database tries to connect to the primary database. When the end of local log files is reached, the standby database enters a remote catchup pending state.

- PEER

In a Peer state, log data is shipped directly from the primary's log write buffer to the standby whenever the primary flushes its log pages to disk. The HADR synchronization mode specifies whether the primary waits for the standby to send an acknowledgement message that log data is received. The log pages are always written to the local log files on the standby database. This behavior guards against a crash and allows a file to be archived on the new primary in case of takeover, if it is not archived on the old primary. After they are written to local disk, the received log pages can then be replayed on the standby database. If log spooling is disabled (the default), replay reads logs only from the log receive buffer.

- REMOTE_CATCHUP

In a remote catchup state, the primary database reads log data from its log path or a log archiving method and the log data is sent to the standby database. The primary and standby databases enter the Peer state when the standby database receives all the on-disk log data of the primary database. If you are using the SUPERASYNC synchronization mode, the primary and standby never enter the Peer state. They permanently stay in remote catchup state, which prevents the possibility of blocking primary log writing in the Peer state.

- REMOTE_CATCHUP_PENDING

When a standby enters a remote catchup pending state, if a connection to the primary is not established, the standby waits for a connection. After a connection is established, the standby obtains the primary's current log chain information. This information enables the standby, if a log archive is configured, to retrieve log files from the archive and verify that the log files are valid.

- ▶ PRIMARY_MEMBER_HOST

The local host of the primary member that is processing the log stream.

- ▶ PRIMARY_INSTANCE

The instance name of the primary member that is processing the log stream.

- ▶ PRIMARY_MEMBER

The primary member that is processing the log stream.

- ▶ STANDBY_MEMBER_HOST

The local host of the standby member that is processing the log stream.

- ▶ **STANDBY_INSTANCE**
The instance name of the standby member that is processing the log stream.
- ▶ **STANDBY_MEMBER**
The standby member that is processing the log stream.
- ▶ **HADR_CONNECT_STATUS**
The current HADR connection status of the database. Possible values are:
 - **CONGESTED**
The database is connected to its partner node, but the connection is congested. A connection is congested when the TCP/IP socket connection between the primary-standby pair is still alive, but one end cannot send to the other end. For example, the receiving end is not receiving from the socket connection, resulting in a full TCP/IP send space. The reasons for a network connection being congested include the following ones:
 - The network is being shared by too many resources or the network is not fast enough for the transaction volume of the primary HADR node.
 - The server on which the standby HADR node is on is not powerful enough to retrieve information from the communication subsystem at the necessary rate.
 - **CONNECTED**
The database is connected to its partner node.
 - **DISCONNECTED**
The database is not connected to its partner node.
- ▶ **HADR_CONNECT_STATUS_TIME**
The time when the current HADR connection status began. Depending on the HADR_CONNECT_STATUS value, the HADR_CONNECT_STATUS_TIME value indicates:
 - Congestion start time.
 - Connection start time.
 - Disconnection time.
- ▶ **HEARTBEAT_INTERVAL**
The heartbeat interval in seconds, which is computed from various factors, such as the values of the `hadr_timeout` and `hadr_peer_window` configuration parameters. The HEARTBEAT_INTERVAL element indicates how often the primary and standby exchange monitor information.

- ▶ **HADR_TIMEOUT**

The time, in seconds, by which a HADR database must receive a message from its partner database. After this period, a HADR database server considers that the connection between the databases failed and disconnects.

- ▶ **TIME_SINCE_LAST_RECV**

The time, in seconds, that elapsed since the last message was received, so the larger the number, the longer the delay in message delivery. When the TIME_SINCE_LAST_RECV value equals the HADR_TIMEOUT value, the connection between the databases is closed.

- ▶ **PEER_WAIT_LIMIT**

The length of time, in seconds, that the primary database waits before it breaks out of the Peer state if logging is blocked while it waits for HADR log shipping to the standby. A value of 0 indicates no timeout.

- ▶ **LOG_HADR_WAIT_CUR**

The length of time, in seconds, that the logger waits on a HADR log shipping request. A value of 0 is returned if the logger is not waiting. When the wait time reaches the value that is returned in the PEER_WAIT_LIMIT field, HADR breaks out of the Peer state to unblock the primary database.

- ▶ **LOG_HADR_WAIT_RECENT_AVG**

The average time, in seconds, for each log flush. This monitoring information is only reported by the **db2pd** command.

- ▶ **LOG_HADR_WAIT_ACCUMULATED**

The accumulated time, in seconds, that the logger spent waiting for HADR to ship logs. This monitoring information is only reported by the **db2pd** command.

- ▶ **LOG_HADR_WAITS_COUNT**

The total count of HADR wait events in the logger. The count is incremented every time the logger initiates a wait on HADR log shipping, even if the wait returns immediately. As a result, this count is effectively the number of log flushes while the databases are in the Peer state. This monitoring information is only reported by the **db2pd** command.

- ▶ **LOG_HADR_WAIT_TIME**

The accumulated time, the logger spent, waiting for HADR to ship logs. With LOG_HADR_WAIT_TIME and LOG_HADR_WAITS_TOTAL, you can compute the average HADR wait time per log flush in arbitrary interval. The units are milliseconds. This monitoring information is reported only by the **MON_GET_HADR** table function.

- ▶ **LOG_HADR_WAITS_TOTAL**

Total count of HADR wait events in the logger. The count is incremented every time the logger initiates a wait on HADR log shipping, even if the wait returns immediately. This count is effectively the number of log flushes in the Peer state. With LOG_HADR_WAIT_TIME and LOG_HADR_WAITS_TOTAL, you can compute the average HADR wait time per log flush in an arbitrary interval. This monitoring information is reported only by the **MON_GET_HADR** table function.
- ▶ **SOCK_SEND_BUF_REQUESTED,ACTUAL**

The requested socket send buffer size (SOCK_SEND_BUF_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used). The actual socket send buffer size (SOCK_SEND_BUF_ACTUAL), in bytes.
- ▶ **SOCK_RECV_BUF_REQUESTED,ACTUAL**

The requested socket receive buffer size (SOCK_RECV_BUF_REQUESTED), in bytes. A value of 0 indicates no request (the system default is used). The actual socket receive buffer size (SOCK_RECV_BUF_ACTUAL), in bytes.
- ▶ **PRIMARY_LOG_FILE,PAGE,POS**

The name of the current log file of the log stream on the primary database (PRIMARY_LOG_FILE). The page number in the current log file that indicates the current log position on the primary HADR database. The page number is relative to its position in the log file. For example, page 0 is the beginning of the file (PRIMARY_LOG_PAGE). The current receive log position (byte offset) of the log stream on the primary database (PRIMARY_LOG_POS).
- ▶ **STANDBY_LOG_FILE,PAGE,POS**

The name of the log file corresponding to the standby receive log position on the log stream (STANDBY_LOG_FILE). The page number (relative to its position in the log file) corresponding to the standby receive log position (STANDBY_LOG_PAGE). The current log position of the standby HADR database (STANDBY_LOG_POS).
- ▶ **HADR_LOG_GAP**

The running average, in bytes, of the gap between the PRIMARY_LOG_POS value and STANDBY_LOG_POS value.

- ▶ STANDBY_REPLAY_LOG_FILE,PAGE,POS
The name of the log file corresponding to the standby replay log position on the log stream (STANDBY_REPLAY_LOG_FILE). The page number in the standby replay log file corresponding to the standby replay log position (STANDBY_REPLAY_LOG_PAGE). The page number is relative to its position in the log file. For example, page 0 is the beginning of the file. The byte offset of the standby replay log position on the log stream (STANDBY_REPLAY_LOG_POS).
- ▶ STANDBY_RECV_REPLY_GAP
The average, in bytes, of the gap between the standby log receive position and the standby log replay position. If the value of this gap reaches the combined value of the standby's receive buffer size and the standby's spool limit, the standby stops receiving logs and blocks the primary if it is in the Peer state.
- ▶ PRIMARY_LOG_TIME
The latest transaction time stamp of the log stream on the primary database.
- ▶ STANDBY_LOG_TIME
The latest transaction time stamp of received logs on the log stream on the standby database.
- ▶ STANDBY_REPLY_LOG_TIME
The transaction time stamp of logs being replayed on the standby database.
- ▶ STANDBY_RECV_BUF_SIZE
The standby receive buffer size, in pages.
- ▶ STANDBY_RECV_BUF_PERCENT
The percentage of the standby log receive buffer that is being used. Even if this value is 100, indicating that the receive buffer is full, the standby can continue to receive logs if you enabled log spooling.
- ▶ STANDBY_SPOOL_LIMIT
The maximum number of pages to spool. A value of 0 indicates that log spooling is disabled; a value of -1 indicates that there is no limit.
- ▶ PEER_WINDOW
The value of the `hadr_peer_window` database configuration parameter.
- ▶ READS_ON_STANDBY_ENABLED
An indicator of whether the HADR reads on standby feature is enabled.

Possible values are:

- Y
 - N
- STANDBY_REPLAY_ONLY_WINDOW_ACTIVE
- An indicator of whether the replay-only window (caused by DDL or maintenance-operation replay) is in progress on the standby, meaning that readers are not allowed on the standby. Possible values are:
- Y
 - N
- PEER_WINDOW_END
- The point in time until which the primary database stays in a Peer or disconnected Peer state, if the primary database is active. The field is displayed only if you enabled a peer window.
- STANDBY_REPLAY_DELAY
- Indicates the value of the `hadr_replay_delay` database configuration parameter.
- TAKEOVER_APP_REMAINING_PRIMARY
- The current number of applications still to be forced off the primary during a non-forced takeover. This field is displayed only if there is a non-forced takeover in progress.
- TAKEOVER_APP_REMAINING_STANDBY
- The current number of applications still to be forced off the read-enabled standby during a takeover. This field is displayed only if there is a takeover in progress.
- STANDBY_REPLAY_ONLY_WINDOW_START
- The time at which the current replay-only window became active. This field is displayed only if there is an active replay-only window on the read-enabled standby.
- STANDBY_REPLAY_ONLY_WINDOW_TRAN_COUNT
- The total number of existing uncommitted DDL or maintenance transactions that are executed in the current replay-only window. This field is displayed only if there is an active replay-only window on the read-enabled standby.
- INACTIVESTANDBY_REPLAY_ONLY_WINDOW_START
- Replay only window start time. This monitoring information is only reported by the `MON_GET_HADR` table function.



DB2 and system upgrades

In this chapter, we focus on applying fix packs and version upgrades to DB2 High Availability Disaster Recovery (HADR) databases. Fix packs can be applied with no downtime and DB2 upgrades can be applied with minimal downtime. We also explore important considerations that relate to any changes to other system components that might require DB2 or the whole server to be restarted. Lastly, we provide an overview of what to consider when you update certain database configuration parameters.

We cover the following topics with techniques for GUI and command-line interfaces to cater for both styles of DB2 administration on platforms that are supported by the DB2 Data Server. We use IBM Data Studio as the GUI where appropriate.

This chapter covers the following topics:

- ▶ General steps for upgrades in a HADR environment
- ▶ DB2 fix pack rolling upgrades
- ▶ DB2 upgrade
- ▶ Rolling operating system and DB2 configuration parameter updates

9.1 General steps for upgrades in a HADR environment

Here are the general steps to perform a rolling fix pack or version upgrade where the HADR Primary database is never taken offline and the users experience no downtime. The specific examples that are presented in this section include a Primary and Principle Standby, which can be extended only by following these general steps.

The procedure is essentially the same as with single standby mode, except you should perform the upgrade on one database at a time and start with an auxiliary standby. For example, consider the following HADR setup:

- ▶ node1 is the primary.
- ▶ node2 is the principal standby.
- ▶ host 3 is the auxiliary standby.

For this setup, perform the rolling upgrade or update by completing the following steps:

1. Deactivate node3, make the required changes, activate node3, and start HADR on node3 (as a standby).
2. After node3 is caught up in log replay, deactivate node2, make the required changes, activate node2, and start HADR on node2 (as a standby).
3. After node2 is caught up in log replay and in the Peer state with node1, run a takeover on node2.
4. Deactivate node1, make the required changes, activate node1, and start HADR on node1 (as a standby).
5. After node1 is in the Peer state with host 2, run a takeover on node1 so that it becomes the primary again and node2 becomes the principal standby again.

9.2 DB2 fix pack rolling upgrades

HADR gives you high availability while you apply DB2 fix packs through rolling upgrades. Your database downtime is only minimal while you switch roles between your database servers. There is no loss of data and no visible downtime to clients. This situation assumes that your application uses standard retry logic and handles the reconnect error message (SQL30108N) if you are using automatic client reroute (ACR).

In this section, to illustrate the fix pack rolling upgrade procedure through IBM Data Studio, we provide an example of the application of a rolling upgrade fix pack on a HADR setup that involves two Linux servers.

9.2.1 Rolling upgrade on Linux

In this section, we describe the procedure to apply a DB2 fix pack with no downtime on Linux, from DB2 AESE 10.1 fix pack 0 to fix pack 1, using IBM Data Studio wherever practical.

IBM Data Studio: IBM Data Studio provides database developers and DBAs with an integrated, modular environment for productive administration of DB2 for Linux, UNIX, and Windows. It also provides collaborative database development tools for DB2, Informix, Oracle, and Sybase. It can be downloaded at no cost at:

<http://www.ibm.com/developerworks/downloads/im/data/>

We use two servers, node1 and node2. Both have the database SAMPLE, as HADR Primary on node1, and as HADR Standby on node2. Our DB2 instance name is the default “DB2”. The DB2 instance user ID is db2inst1.

To apply the fix pack, complete the following steps:

1. Extract the DB2 fix pack in to a source location on both servers. This action minimizes the time that the HADR Standby is down and confirms the integrity of the fix pack download. We extract our fix pack to /usr/tmp/aese on both servers by running a command similar to Example 9-1 on both servers.

Example 9-1 Extract the fix pack

```
db2inst1@node1:/usr/tmp> tar -xvf v10.1fp1_linuxx64_aese.tar.gz
```

2. Confirm the prerequisite status. The HADR pair should be in the Peer state before you start the rolling upgrade. The current HADR state can be checked by running **db2pd -d <dbname> -hadr** (Example 9-2).

Example 9-2 Check the HADR state

```
db2inst1@node1:~> db2pd -d sample -hadr
```

```
Database Member 0 -- Database SAMPLE -- Active -- Up 9 days 23:23:21  
-- Date 21/07/2012 12:13:17
```

HADR_ROLE = PRIMARY

```

        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = NEARSYNC
        STANDBY_ID = 1
        LOG_STREAM_ID = 0
        HADR_STATE = PEER
PRIMARY_MEMBER_HOST = node1
        PRIMARY_INSTANCE = db2inst1
        PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = node2
        STANDBY_INSTANCE = db2inst1
        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 21/07/2012 12:12:50.120336
(1342869170)
        HEARTBEAT_INTERVAL(seconds) = 30
        HADR_TIMEOUT(seconds) = 120
        TIME_SINCE_LAST_RECV(seconds) = 26
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000027
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.738
        LOG_HADR_WAIT_COUNT = 26493
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000007.LOG, 890, 72920803
        STANDBY_LOG_FILE,PAGE,POS = S0000007.LOG, 890, 72920803
        HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000007.LOG, 890, 72920803
        STANDBY_RECV_REPLY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 21/07/2012 12:05:06.000000
(1342868706)
        STANDBY_LOG_TIME = 21/07/2012 12:05:06.000000
(1342868706)
        STANDBY_REPLY_LOG_TIME = 21/07/2012 12:05:06.000000
(1342868706)
        STANDBY_RECV_BUF_SIZE(pages) = 512
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 0
        PEER_WINDOW(seconds) = 0
        READS_ON_STANDBY_ENABLED = N

```

3. Apply the DB2 fix pack code on the standby server (node2 in our case). For more information about the actual fix pack application, see the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db21uw/v10r1/topic/com.ibm.db2.1uw.qb.server.doc/doc/t0006352.html>

The basic fix pack installation steps for DB2 AESE 10.1 fix pack 1 are as follows:

- a. Deactivate your standby database and stop the DB2 instance. From IBM Data Studio, expand the object tree down to the database object. Right-click the database name, and then click **Stop**. This action stops the instance and the database. Figure 9-1 shows that the database is deactivated.

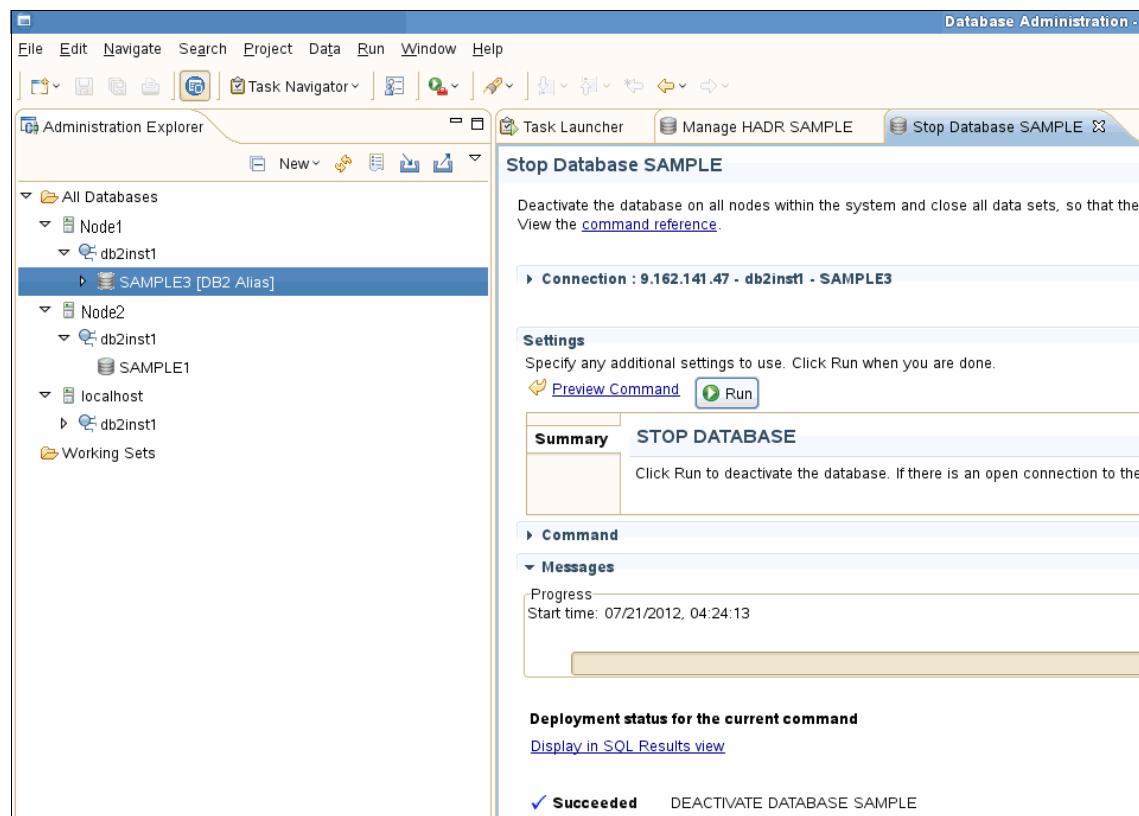


Figure 9-1 Run the command to stop the databases and the instance

- b. Open a terminal session and open the folder where the extracted fix pack files are located. For our DB2 10.1 ESE fp1, the fix pack is installed as shown in Example 9-3. The text in bold is entered during the fix pack installation. For practical reasons and simplicity, some of the output in Example 9-3 is removed (marked as “**<removed>**”).

Example 9-3 Install the fix pack

```
db2inst1@node2:~> cd /usr/tmp/aese  
db2inst1@node2:~> ./installFixPack /opt/ibm/db2/V10.1/  
node2:/usr/tmp/aese # ./installFixPack /opt/ibm/db2/V10.1/
```

Enter the full path of the base installation directory:

```
-----  
/opt/ibm/db2/V10.1/
```

Do you want to choose a different installation directory for the fix pack? [yes/no]

```
-----  
-----  
no
```

```
DBI1017I installFixPack is updating the DB2 product(s) installed  
in  
location /opt/ibm/db2/V10.1/.
```

DB2 installation is being initialized.
Total number of tasks to be performed: 40
Total estimated time for all tasks to be performed: 1778
second(s)

Task #1 start
Description: Stopping DB2 Fault Monitor
Estimated time 10 second(s)
Task #1 end

<removed>

Task #41 start
Description: Updating existing DB2 instances
Estimated time 60 second(s)
Task #41 end

The execution completed successfully.

For more information see the DB2 installation log at
"/tmp/installFixPack.log.7037".

- c. Activate the HADR Standby database by running the command that is shown in Example 9-4 on the command line.

Example 9-4 Activate database command

```
db2inst1@node2:~> db2 activate db sample
```

- d. Confirm that HADR is in a Connected Peer state by running the command that is shown in Example 9-5.

Example 9-5 Checking the HADR Peer state

```
db2inst1@node2:~> db2pd -d sample -hadr
```

```
Database Member 0 -- Database SAMPLE -- Standby -- Up 0 days
00:00:44 -- Date 2012-07-22-15.55.36.098555
```

```

          HADR_ROLE = STANDBY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = NEARSYNC
          STANDBY_ID = 0
          LOG_STREAM_ID = 0
          HADR_STATE = PEER
          PRIMARY_MEMBER_HOST = node1
          PRIMARY_INSTANCE = db2inst1
          PRIMARY_MEMBER = 0
          STANDBY_MEMBER_HOST = node2
          STANDBY_INSTANCE = db2inst1
          STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = CONNECTED
          HADR_CONNECT_STATUS_TIME = 22/07/2012
15:54:53.084960 (1342968893)
          HEARTBEAT_INTERVAL(seconds) = 30
          HADR_TIMEOUT(seconds) = 120
          TIME_SINCE_LAST_RECV(seconds) = 12
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
          LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000027
          LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.738
          LOG_HADR_WAIT_COUNT = 26493
```

```
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
    PRIMARY_LOG_FILE,PAGE,POS = S0000008.LOG, 509,
75444907
    STANDBY_LOG_FILE,PAGE,POS = S0000008.LOG, 509,
75444907
        HADR_LOG_GAP(bytes) = 0
    STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000008.LOG, 509,
75444907
        STANDBY_RECV_REPLY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 22/07/2012
15:05:17.000000 (1342965917)
        STANDBY_LOG_TIME = 22/07/2012
15:05:17.000000 (1342965917)
        STANDBY_REPLY_LOG_TIME = 22/07/2012
15:05:17.000000 (1342965917)
        STANDBY_RECV_BUF_SIZE(pages) = 512
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 0
        PEER_WINDOW(seconds) = 0
        READS_ON_STANDBY_ENABLED = N
```

4. Switch HADR roles.

From IBM Data Studio, expand the object tree down to the database object. Right-click the database name, and then click **Manage HADR** (Figure 9-2).

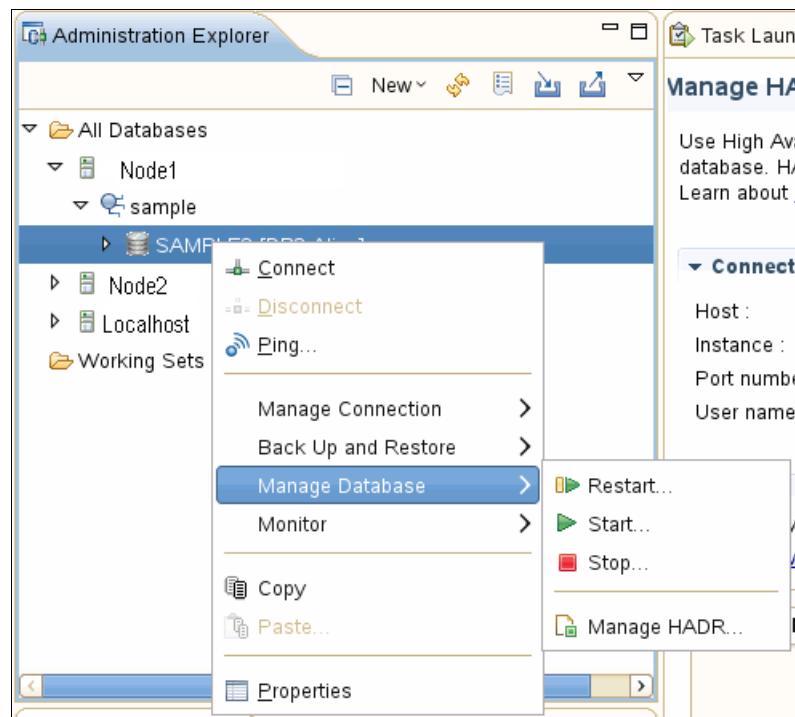


Figure 9-2 Select Manage HADR for the Standby database

- a. Select **Manage Standby database** and **Takeover a primary** (Figure 9-3).

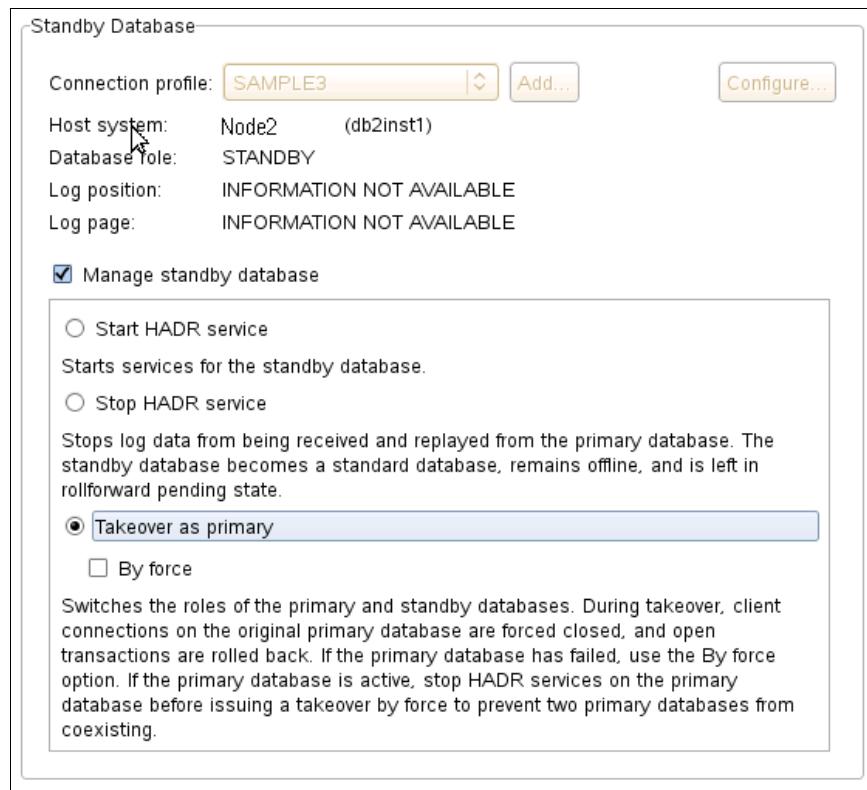


Figure 9-3 Taking over as the primary

- b. To confirm the takeover, click **Command** and review the code to be run and then click **Run** (Figure 9-4).

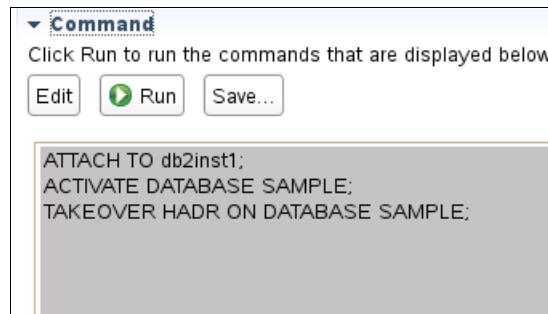


Figure 9-4 Confirm and run the commands

- c. A DB2 message should appear and confirm success.
- d. The Manage High Availability Disaster Recovery (HADR) panel should now show that the standby and primary roles switched servers. In our example in Example 9-6, node2 is now the primary.

Example 9-6 Confirm a successful takeover

```
db2inst1@node2:~> db2pd -db sample -hadr

Database Member 0 -- Database SAMPLE -- Active -- Up 0 days
00:27:17 -- Date 2012-07-22-16.22.09.700721

          HADR_ROLE = PRIMARY
          REPLAY_TYPE = PHYSICAL
          HADR_SYNCMODE = NEARSYNC
          STANDBY_ID = 1
          LOG_STREAM_ID = 0
          HADR_STATE = DISCONNECTED
PRIMARY_MEMBER_HOST = node2
PRIMARY_INSTANCE = db2inst1
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = node1
STANDBY_INSTANCE = db2inst1
STANDBY_MEMBER = 0
          HADR_CONNECT_STATUS = DISCONNECTED
          HADR_CONNECT_STATUS_TIME = 22/07/2012
16:06:40.173853 (1342969600)
          HEARTBEAT_INTERVAL(seconds) = 30
          HADR_TIMEOUT(seconds) = 120
TIME_SINCE_LAST_RECV(seconds) = 0
          PEER_WAIT_LIMIT(seconds) = 0
          LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
          LOG_HADR_WAIT_COUNT = 0
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
          PRIMARY_LOG_FILE,PAGE,POS = S0000008.LOG, 510,
75447567
          STANDBY_LOG_FILE,PAGE,POS = S0000000.LOG, 0, 0
          HADR_LOG_GAP(bytes) = 0
STANDBY_REPLY_LOG_FILE,PAGE,POS = S0000000.LOG, 0, 0
          STANDBY_RECV_REPLY_GAP(bytes) = 0
          PRIMARY_LOG_TIME = 22/07/2012
16:08:36.000000 (1342969716)
```

```
STANDBY_LOG_TIME = NULL  
STANDBY_REPLAY_LOG_TIME = NULL  
PEER_WINDOW(seconds) = 0
```

HADR supports a standby that runs a newer DB2 level than the primary, but not the other way around. Therefore, when the takeover is complete, HADR is in the 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 3 on page 357 on the new standby server. In our example, we perform the fix pack application on node1.
 - b. Post-installation tasks such as package binding for DB2 CLI cannot be performed from a HADR standby database, so you must connect to the current HADR primary database (in our example, SAMPLE on node2). When connected, run the binds appropriate to your environment, as described in the DB2 10.1 Information Center at:
<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.qb.server.doc/doc/c0025015.html>
- The binds and system catalog updates are logged, so HADR replays them on the standby to complete the fix pack application.
6. (Optional) Switch the roles back to the original primary if you want to restore your original server roles.
 - a. The HADR status must be a Connected Peer state to ensure that all logs are applied.
 - b. Repeat step 3 on page 357 to run an unforced HADR takeover.

9.3 DB2 upgrade

A rolling upgrade is not supported between DB2 version or major subversion upgrades, such as Version 9.1 to 9.5 or 9.7 to 10.1. For these events, plan for a database outage while the HADR primary database is updated. In this section, we provide an example of a DB2 upgrade on a Linux operating system, using IBM Data Studio (where practical) and the CLI.

DB2 installations and upgrades have improved in recent versions by containing the base code for an installation in the same downloadable package as each fix pack. This setup reduces downtime, and greatly simplifies installations and upgrades. Having everything in a single package also avoids many situations where post-installation steps are neglected.

For your own reference, documentation about related concepts that are involved in DB2 upgrades can be found in the DB2 10.1 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.qb.upgrade.doc/doc/c0023662.html>

The initial critical step of DB2 upgrades is running **db2ckupgrade**. The command checks if the databases are eligible for an upgrade to the new DB2 version. The **db2ckupgrade** command listing dependencies for success are covered in the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002055.html>

9.3.1 DB2 version upgrade on Linux

Here we describe a step-by-step upgrade from DB2 Enterprise Server Edition Version 9.7 fix pack 0 to Version 10.1 fix pack 0, on a pair of Linux servers. Our example HADR database pair name is SAMPLE.

All steps on Linux are performed by the DB2 instance user (db2inst1 for our example). We illustrate all the dialog boxes for the DB2 10.1 installation on the HADR standby server only, as they are the same dialog boxes we see on the HADR primary server.

To help minimize downtime on the HADR primary server, extract the DB2 compressed package on to both servers before you stop any DB2 instances. Extract the installation files to a target directory (/usr/tmp/aese, in our example) to be ready to run **db2setup**, and **db2ckupgrade**. Use the following commands to extract the files:

- ▶ `/usr/tmp/aese/db2setup`
- ▶ `/usr/tmp/aese/avalanche/pureScale/98_FP3_Build_26652/ese_dsf/db2/linux_amd64/utilities/db2ckupgrade`

Do this task beforehand because the extraction process of almost 2000 files normally takes several minutes to perform. Also, this task must be done so you can run **db2ckupgrade** on the DB2 9.7 database to confirm that the migration to DB2 10.1 works without more structural changes.

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.

Complete the following steps:

1. Upgrade the standby server.

- a. Ensure that your HADR server is in the Connected Peer state by running **db2pd get snapshot**, or use the Manage HADR window in IBM Data Studio. For details about this command's usage, see Chapter 8, "HADR monitoring" on page 337.

Example 9-7 shows the **db2pd** command output for servers node1 and node2.

Example 9-7 HADR Connected Peer state before upgrade

```
db2inst2@node1:~> db2pd -db sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days
03:40:58 -- Date 07/24/2012 10:29:22

HADR Information:
Role      State           SyncMode HeartBeatsMissed
LogGapRunAvg (bytes)
Primary   Peer            Nearsync 0                      273

ConnectStatus ConnectTime               Timeout
Connected    Tue Jul 24 06:48:28 2012 (1343108908) 120

LocalHost          LocalService
node1              60010

RemoteHost         RemoteService
RemoteInstance
node2              60010                  db2inst2

PrimaryFile PrimaryPg PrimaryLSN
S0000001.LOG 621      0x0000000002961C62

StandByFile StandByPg StandByLSN
S0000001.LOG 621      0x0000000002961C22
```

db2pd output: The output from **db2pd** shown in Example 9-7 on page 366 is rather different from the earlier examples of output from **db2pd**. The command is being run against DB2 9.7, and the earlier examples were run against DB2 10.1. Some format changes were made to **db2pd** between Version 9.7 and Version 10.1.

- b. Run **deactivate database** on the CLP:

```
db2inst2@node2:~> db2 deactivate db sample  
DB20000I The DEACTIVATE DATABASE command completed successfully.
```

- c. You can reduce downtime further by checking your database structure and content for eligibility before the upgrade downtime window by running **db2prereqcheck** from the extracted installation sources of the new DB2 version. This action avoids added downtime on the primary database in later steps, at the cost of some added work before downtime.

Before you run **db2prereqcheck**, you must:

- i. Stop HADR on the standby database by running the following command:

```
db2inst2@node2:~> 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):

```
db2inst2@node2:~> db2 update dbm cfg using svcename 99999  
db2inst2@node2:~> db2stop force  
db2inst2@node2:~> db2start
```

Now you can run **db2ckupgrade** from the extracted subdirectory:

```
db2ckupgrade sample
```

If the output is satisfactory, you can proceed to step d on page 368; otherwise, corrective action must be taken on the HADR primary database to make it eligible for upgrade.

After corrective action is taken, either run **db2ckupgrade** on the HADR primary, or proceed with the following steps for the HADR standby if you want to avoid further downtime. Use the command line processor (CLP):

- i. Perform an online database backup of HADR Primary with included logs by running the following command:

```
db2inst2@node1:~> db2 backup db sample online to  
/usr/tmp/backup include logs
```

- ii. Copy the resulting database backup file to the standby server by running the following command:

```
db2inst2@node1:/usr/tmp/backup> scp  
.SAMPLE.0.db2inst2.NODE0000.CATN0000.20120724150616.001  
root@node2:/usr/tmp/backup
```

- iii. Restore the database on the standby server, specifying a destination for the included log files and roll the database forward (Example 9-8).

Example 9-8 Restoring a database from a backup

```
db2inst2@node2:~> db2 restore db sample from /usr/tmp/backup
taken at 20120724112114 logtarget /usr/tmp replace history
file
SQL2539W Warning! Restoring to an existing database that is
the same as the backup image database. The database files be
deleted.
Do you want to continue ? (y/n) y
DB20000I The RESTORE DATABASE command completed successfully.

db2 => rollforward database sample to end of logs and complete
overflow log path (/usr/tmp/)
```

Rollforward Status

Input database alias	= sample
Number of nodes have returned status	= 1
Node number	= 0
Rollforward status	= not pending
Next log file to be read	=
Log files processed	= S0000001.LOG -
S0000001.LOG	
Last committed transaction	=
2012-07-24-14.22.10.000000 UTC	

```
DB20000I The ROLLFORWARD command completed successfully.
```

Do not run **db2ckupgrade** yet because the **db2ckupgrade** command cannot be run on a database while it is in HADR standby or rollforward pending mode. Also, replace it with a backup of the HADR primary after you upgrade the primary to a DB2 10.1 database structure.

- d. If you have not done so, stop the DB2 instance from the CLP or IBM Data Studio, as shown in Example 9-9.

Example 9-9 Stop the DB2 instance from the CLP

```
db2inst2@node1:/usr/tmp/backup> db2 stop hadr on db sample
db2inst2@node1:/usr/tmp/backup> db2stop
```

- e. Close the IBM Data Studio and any other DB2 GUI tools or windows.

- f. Perform the DB2 10 installation.

Perform the installation of the DB2 Enterprise Server edition software but do not create an instance during this process. For detailed steps about how to perform this installation, go to the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.qb.server.doc/doc/t0059589.html>

- g. Perform the instance upgrade.

As shown in the Example 9-10, you can upgrade the instance on the standby by running **db2iupgrade** in the new code base.

Example 9-10 Upgrade the instance

```
db2inst2@node2:~> cd /opt/ibm/db2/V10.1/instance
db2inst2@node2:~> /db2iupgrade
DBI1446I The db2iupgrade command is running.
DB2 installation is being initialized.
Total number of tasks to be performed: 4
Total estimated time for all tasks to be performed: 309 second(s)
Task #1 start
Description: Setting default global profile registry variables
Estimated time 1 second(s)
Task #1 end
Task #2 start
Description: Initializing instance list
Estimated time 5 second(s)
Task #2 end
Task #3 start
Description: Configuring DB2 instances
Estimated time 300 second(s)
Task #3 end
Task #4 start
Description: Updating global profile registry
Estimated time 3 second(s)
Task #4 end
The execution completed successfully.
For more information see the DB2 installation log at
"/tmp/db2iupgrade.log.31440".
Required: Review the following log file also for warnings or
errors:
"/tmp/db2iupgrade_local.log.*"
DBI1070I Program db2iupgrade completed successfully.
```

- h. Confirm that the new version is installed.

Confirm your new DB2 version by running **db2level** (Example 9-11).

Example 9-11 Check the DB2 version

```
db2inst2@node2:~> db2level
DB21085I This instance or install (instance name, where
applicable:
"db2inst2") uses "64" bits and DB2 code release "SQL10011" with
level
identifier "0202010E".
Informational tokens are "DB2 v10.1.0.1", "s120715", "IP23384",
and Fix Pack
"1".
Product is installed at "/opt/ibm/db2/V10.1".
```

2. Upgrade the primary server:

Log on to the primary server (node1 in our case) with the DB2 instance user. Because of the different version levels of your two servers, you cannot switch roles to the standby, so your database downtime for users starts here.

Complete the following steps:

- a. Arrange for all DB2 connections (users, applications, and batch) to stop. This action means that you do not need to run **force applications**, which can cause potentially lengthy rollback actions. Removal of connections can be measured with by running **db2 list applications** from the CLP.

- b. From the CLP, run **deactivate database** as follows:

```
db2inst2@node1:~> db2 deactivate database sample
```

The **deactivate database** command puts database into a consistent state in preparation for the migration check and later structural conversion.

- c. Perform a backup of your database (use offline mode for ease of restoration without logs or rolling forward). This backup can be completed either through IBM Data Studio or CLP. 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 remigration to the new DB2 version is practical. Run the following command:

```
db2inst2@node1:~> db2 backup database sample to /usr/tmp/backup
```

- d. Run **db2support** with collection level zero to get instance configuration information and other information for fallback if necessary:

```
db2inst2@node1:~> db2support /usr/tmp -c1 0
```

Run **db2cfexp** to save node and database directory information if the DB2 product code must be uninstalled and reinstalled:

```
db2inst2@node1:~> db2cfexp /usr/tmp/nodedb.cfg backup
```

After any fallback requiring reinstallation of DB2 product code, the **db2cfimp** command can be used to reimport these definitions where necessary:

```
db2inst2@node1:~> db2cfimp /usr/tmp/nodedb.cfg
```

- e. Perform steps 1.d on page 368, through to 1.h on page 370 to install the new DB2 version on the HADR primary server.

You can perform step f on page 369 to install the DB2 sources beforehand to save time.

The **db2iupgrade** command implicitly calls **db2ckupgrade** on all databases in that instance, and does not migrate the instance if any errors are found. This situation emphasizes the need to perform **db2ckupgrade** well in advance to ensure that no failures occur at a more critical stage.

The **db2iupgrade** command is described at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002055.html>

- f. Run **upgrade database** on all local databases in the new DB2 10.1 instances. This command also applies to any system tools database (used by the DB2 Task Scheduler). We migrate our HADR database by running the following command:

```
db2inst2@node1:~> db2 upgrade database sample
```

Successful completion of this command results in the following message:
DB20000I The UPGRADE DATABASE command completed successfully.

The **upgrade database** command is described at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0001959.html>

At this stage, you can consider the actual DB2 upgrade process as completed, and application connectivity testing can commence. The next steps can be performed in parallel with user testing before you consider your system ready for production.

3. Re-establish HADR database pairing between the standby server (node2) and the primary server (node1):

- a. Perform an online backup of your HADR primary database, which is copied over to the standby server to re-establish HADR. As in step 2.c on page 370, you can use the Data Studio Backup wizard, or the CLP, as we do here:

```
db2inst2@node1:~> db2 backup database sample online to  
/usr/tmp/backup
```

- b. Copy the backup image over to the HADR standby server that is ready to restore.
 - c. Log on to the standby server and restore the database from the backup image that is copied across in step 3.b. You can either use IBM Data Studio or the **restore** command on the CLP. Our CLP syntax is:

```
db2inst2@node2:~> db2 restore database sample from  
/usr/tmp/backup taken at yyyymmddhhmmss
```

Accept the request to overwrite the existing database. A successful output from this command when run against an existing DB2 8.2 database includes an extra message:

```
SQL2555I The restored database was successfully upgraded to the current release.
```

This message indicates that your pre-existing standby database is upgraded as part of the restore.

- d. Update the database configuration parameters for the HADR standby. Normally, if you restored the primary database over the existing unchanged old HADR standby database, no action would be necessary here, because the existing database configuration parameters for the HADR standby are retained.

We experienced that after our DB2 10.1 migration with restore completed that the database configuration parameters reverted to the HADR primary settings. This situation is why it is imperative to check that the HADR standby database configuration parameters are correct, and update any of the parameters as necessary.

Use this CLP command syntax to get the current settings:

```
db2 get db cfg for sample | grep "HADR"
```

We expect four key HADR primary database and standby database configuration parameters of our example to be as shown in Table 9-1.

Table 9-1 Expected primary and standby HADR db cfg values

Parameter	Primary	Standby
HADR local host name (HADR_LOCAL_HOST)	node1	node2
HADR local service name (HADR_LOCAL_SVC)	60010	60010
HADR remote service name (HADR_REMOTE_HOST)	node2	node1
HADR remote service name (HADR_REMOTE_SVC)	60010	60010

Our standby **db cfg** values need to be reset by running the following commands:

- **db2 update db cfg for sample using hadr_local_host node2**
 - **db2 update db cfg for sample using hadr_local_svc 60010**
 - **db2 update db cfg for sample using hadr_remote_host node1**
 - **db2 update db cfg for sample using hadr_remote_svc 60010**
- e. If you disabled remote user access in step 1.c on page 367 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 by running the following commands:
- **db2 update dbm cfg using svcename 50000**
 - **db2stop**
 - **db2start**
- f. Start HADR on the 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 367 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 also has the STANDARD role, and we change it to the PRIMARY role by running another **start hadr**.

Commands to start HADR can be run remotely through the CLP or IBM Data Studio if remote database aliases cataloged for the other database in the HADR pair. Start HADR on the standby first, then the primary, which is what IBM Data Studio always does, and what you should also do when you use the CLP unless you can ensure that you are able to start the standby within the **HADR_TIMEOUT** value after you start the primary.

We present the following examples for your reference, so you can choose the technique that you prefer:

- i. Using CLP on the standby server by running the following commands:

- **db2 deactivate db sample**
- **db2 start hadr on db sample as standby**
- **db2 deactivate db samppug user db2inst2 using *passblah***
- **db2 start hadr on db samppug user db2inst2 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 node1 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 the user and password.

- ii. If you use CLP on the primary server, run the following commands:

- **db2 deactivate db sampfar user db2inst2 using *passblah***
- **db2 start hadr on db sampfar user db2inst2 using *passblah* as standby**
- **db2 deactivate db sample**
- **db2 start hadr on db sample as primary**

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 node2 server.

HADR should now be started on the primary and standby servers, and in the Connected Peer state within a 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 that DB2 has a working hot backup with the HADR standby database.

There is a reason to start HADR on the standby first. A Standard role database does not normally activate in the HADR primary role without a HADR standby being activated and successfully communicating within the HADR_TIMEOUT period. A HADR standby database can be started without a HADR primary being present or active. An eligible Standard role database can be started only as a HADR primary without a connection to an active HADR Standby by 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 task can fail with error code SQL1767N or SQL1768N, depending on the initial state of the database. This situation is why the **deactivate database** command precedes all **start hadr** commands that are run by IBM Data Studio. 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 10.1 Information Center contains a useful table about how a database in any state behaves when a **start hadr** command is issued against it:

http://publib.boulder.ibm.com/infocenter/db21uw/v10r1/topic/com.ibm.db2._luw.admin.cmd.doc/doc/r0011551.html

Here is the equivalent link for **stop hadr**:

http://publib.boulder.ibm.com/infocenter/db21uw/v10r1/topic/com.ibm.db2._luw.admin.cmd.doc/doc/r0011552.html

9.4 Rolling operating system and DB2 configuration parameter updates

This section covers high-level instructions for updating database and database manager configuration parameters, and for any operating system (OS) and application upgrades. To keep your HADR systems optimal, you should apply changes to both servers as quickly as possible.

When you update HADR-specific database configuration parameters, DB2 does not allow you to switch HADR roles. Certain HADR parameters must always remain matched, such as **HADR_TIMEOUT**. You must have a short database outage while you update 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 that 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 that are dynamic are listed under the column heading “Cfg Online” at the DB2 10.1 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.config.doc/doc/r0005181.html>

After the dynamic database manager or database configuration parameter update is issued on the HADR primary, it should be manually issued against the HADR standby as well, because parameter configurations are not logged and not mirrored through HADR log shipping.

9.4.1 Procedure

In our procedure, we use the example environment with node1 and node2. The DB2 database on node1 is initially the HADR primary and node2 is the standby.

Complete the following steps:

1. On node2 in the HADR standby role:
 - a. Ensure that your HADR is in a Peer state by using **db2pd**, a snapshot, or the HADR Manage window.
 - b. Run **deactivate database**.
 - c. If necessary, stop the DB2 Instance, which is necessary for OS or non DB2 changes that require a server recycle, in addition to database manager configuration parameters that 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. Run **activate database** or explicitly connect to the database.
 - g. Check HADR to ensure that the server is in the Peer state by using **db2pd**, a snapshot, or the HADR Manage window.
 - h. Switch the roles of the primary and standby by running an unforced **takeover hadr** command against the standby database.
 - i. Redirect clients to the HADR primary database on node2 through ACR or an IP address switch.
2. On node1 in HADR Standby role, repeat steps 1a to i, but run them against node1.



Automatic client reroute

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

This chapter covers the following topics:

- ▶ ACR overview
- ▶ ACR tuning
- ▶ ACR limitations
- ▶ ACR configuration examples
- ▶ Application programming to handle ACR

For more information about ACR, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.lu.admin.ha.doc%2Fdoc%2Fr0023392.html>

10.1 ACR overview

When there is a loss of communication between a DB2 client and a DB2 (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 fail over to the redundant system when the primary system fails.

ACR 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.

10.1.1 ACR with HADR

Figure 10-1 illustrates the ACR implementation in HADR.

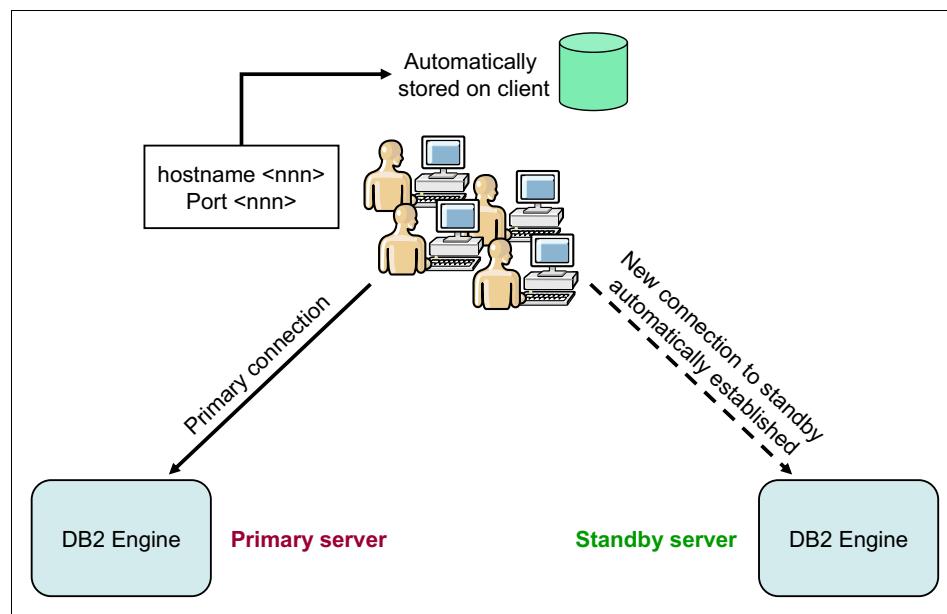


Figure 10-1 ACR implementation in HADR

To enable the ACR feature in HADR, the DB2 server is configured with the name of an alternative location that the DB2 client can access. The **UPDATE ALTERNATE SERVER FOR DATABASE** command is run 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 part of the command. The location is stored in the system database directory of the instance.

The alternate server location information is propagated to the DB2 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.

In general, you can use the ACR feature within the following configurable environments:

- ▶ DB2 Enterprise Server Edition (ESE) with the Data Partitioning Feature (DPF)
- ▶ DB2 Enterprise Server Edition with the IBM DB2 pureScale Feature
- ▶ WebSphere Replication Server
- ▶ IBM PowerHA SystemMirror for AIX
- ▶ High Availability Disaster Recovery (HADR)

10.1.2 ACR in action

Figure 10-2 shows ACR in action when there is a connection failure to a primary database in a HADR environment.

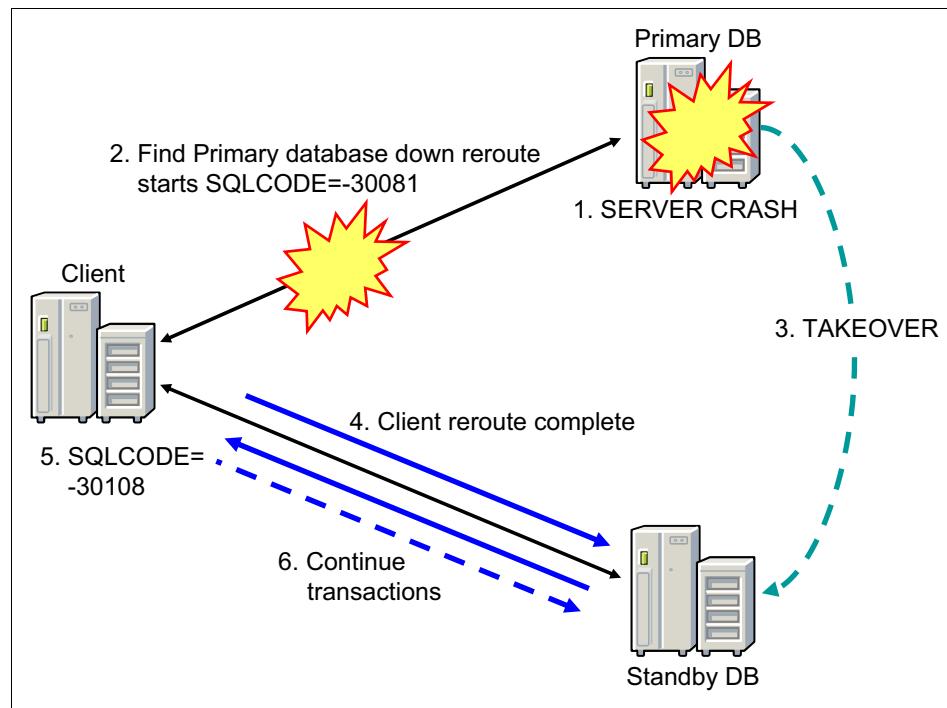


Figure 10-2 ACR implementation in HADR

If an alternate server is specified, ACR is activated when a communication error (SQLCode -30081) is detected or if the DB2 server is not able to accept new requests for any other reason (SQLCode -1224).

When ACR is activated, the 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 to the original server and the alternate server, alternating the attempts between the two servers until it gets a successful connection for 10 minutes from the moment it detected a communication failure. The timing of these attempts varies from the initial rapid connection attempts between the two servers with a gradual lengthening of the intervals between the attempts (Table 10-1).

Table 10-1 ACR connection intervals

Time	Interval between attempts
0 - 30 seconds	0 seconds
30 - 60 seconds	2 seconds
1 minute - 2 minutes	5 seconds
2 - 5 minutes	10 seconds
5 - 10 minutes	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 the application to indicate that a database connection is re-established following the communication failure. The host name or IP address and service name or port number are returned (Example 10-1) for SQLCode -4498.

Example 10-1 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
```

The DB2 client returns only the error for the original communication failure to the application if the re-establishment of the client communication to either the original or alternate server is not possible.

10.2 ACR tuning

DB2 client tries to connect to the original server before it connects to the alternate server when client reroute is triggered. Therefore, to enable fast reroute to the alternate server, the connection to the original server that failed must receive a timeout error quickly. There are three options to achieve this goal:

- ▶ Tuning Internet Protocol network

To make the ACR perform better, you can tune the Internet Protocol network parameters of the operating system on which the DB2 application is running. For example, on the AIX operating system, tune **tcp_keepinit**. The default value 150 means that 75 seconds must pass before the connection is routed to the alternate server.

- ▶ Configuring DB2 registry variables

The operating system network parameters affect all the applications on the system. So, configure the following DB2 client reroute registry variables, which affect only DB2 applications. You must tune the following 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 DB2 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 DB2 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 SQL error code -30081. There is no timeout if the registry variable is not set or is set to 0.

By default, the ACR feature tries 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 tries attempted by the ACR. 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 tries, 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 DB2_MAX_CLIENT_CONNRETRIES or DB2_CONNRETRIES_INTERVAL is not set, the ACR feature reverts to its default behavior of trying the connection repeatedly for up to 10 minutes.

ACR 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 the original server
- Two times for the 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, ACR fails with SQL error code -30081.

For example, assume that the following registry variables are set as follows:

- DB2_MAX_CLIENT_CONNRETRIES=5
- DB2_CONNRETRIES_INTERVAL=3
- DB2TCP_CLIENT_CONTIMEOUT=10

ACR takes $\{(10 \times 4) + 3\} \times 5 = 215$ seconds to give up and returns SQL error code -30081.

Type 4 connectivity: Users of Type 4 connectivity with IBM Data Server Driver for JDBC and SQLJ should use the following two DataSource properties to configure the ACR:

maxRetriesForClientReroute: Use this property to limit the number of tries if the primary connection to the server fails. This property is used only if the retryIntervalForClientReroute property is also set.

retryIntervalForClientReroute: Use this property to specify the amount of time (in seconds) to sleep before it tries 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 it tries to establish a connection to a server before it terminates the connection attempt and generating a communication timeout.

If ACR is activated, you must 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 time out and be rerouted to the alternate server by ACR. 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 db2cli.ini file, you can set the ConnectTimeout CLI/ODBC configuration keyword to specify the time in seconds to wait for a reply when it tries to establish a connection to a server before it terminates the attempt and generating a communication timeout. By default, the client waits indefinitely for a reply from the server when it tries to establish a connection.

If ConnectTimeout is set and ACR 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 you attempt 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 that is specified by the keyword, you receive an error message, as shown in Example 10-2.

Example 10-2 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 you make DB2 database connections through the DB2.NET Provider, you can set the value of the ConnectTimeout property of DB2Connection when you pass the ConnectionString, as shown in Example 10-3.

Example 10-3 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 30 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 a `ConnectTimeout` property for a `ConnectionString` (Example 10-4).

Example 10-4 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 a higher priority than the settings in the registry variables or in the `db2c1i.ini` file.

► Using the same cluster manager

The third option for fast failover is to set up an HADR pair where the primary and standby databases are serviced by the same cluster manager. For more information about this topic, see 10.4.3, “ACR with a HADR database and PowerHA” on page 390. 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 requires an HA cluster configuration.

10.3 ACR limitations

There are some limitations when you use the ACR feature:

- The DB2 server that is installed on the alternate host server must be the same version (but could have a higher or lower fix pack) when compared to the DB2 version 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 must take the alternate server offline and allow the connections to fail back over to the original server. This method requires that the original server be cataloged by running **UPDATE ALTERNATE SERVER** so 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 recatalog 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 re-establish 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 for re-establishing the session resources 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 are 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**
- ▶ Do not run high availability disaster recovery (HADR) commands (**START HADR**, **STOP HADR**, or **TAKEOVER HADR**) on client reroute-enabled database aliases. HADR commands are implemented to identify the target database using database aliases. If the target database has an alternate database that is defined, it is difficult for HADR commands to determine the database on which the command is actually operating. A client might need to connect using a client reroute-enabled alias, but HADR commands must be applied on a specific database. To accommodate this situation, you can define aliases specific to the primary and standby databases and run HADR commands only on those aliases.
- ▶ In a HADR multiple standby setup, you must select one standby database (likely the principal standby) as the alternate server of the primary, because each database server can have only one alternate server defined.

SQL statements: If the client is using CLI, IBM Data Server Driver for JDBC Type 2 or Type 4, and SQLJ drivers, after the connection is re-established, for those SQL statements that are 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.

10.4 ACR configuration examples

ACR 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 (Example 10-5).

Example 10-5 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 is re-established. The transaction that is executing when the communication failure occurs is in an unknown state. The client application must be designed so that it can ignore this error and proceed to the next transaction, or it could try this transaction after it confirms that it did not succeed.

Let us see how ACR works in the following cases:

- ▶ ACR with a non-HADR database
- ▶ ACR with a HADR database
- ▶ ACR with a HADR database and PowerHA

10.4.1 ACR with a non-HADR database

In this example, we use two DB2 host systems, KANAGA and ATLANTIC. At KANAGA, the SAMPLE database is created. Furthermore, the SAMPLE database is also created at the alternate server ATLANTIC with port number 50030.

At the client machine, the SAMPLE database is cataloged at the KANAGA node. The KANAGA node references the KANAGA host name and port 50030.

To activate ACR, you must update the alternate server for SAMPLE database at the KANAGA server as follows:

```
db2 update alternate server for database sample using hostname atlantic
port 50030
```

Without having the ACR feature set up, if there is a communication error when it runs a transaction, the application receives an SQL error code -30081. With the ACR feature set up, DB2 tries to establish the connection to the KANAGA host again. If it is still not working, DB2 tries the alternate server location (the ATLANTIC host 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 failed. Because we do not have HADR in this case, we must 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 SAMPLE database and run a SELECT query, as in Example 10-6, we see that the queries are run at KANAGA.

Example 10-6 Run queries at the 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

1 record(s) selected.
```

If there is a communication error while you run the same query to select one row from the STAFF table, the ACR function routes the query to the alternate server (Example 10-7).

Example 10-7 Connection is rerouted

```
$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

1 record(s) selected.
```

10.4.2 ACR with a HADR database

In this example, we set up a HADR configuration between KANAGA and ATLANTIC for the SAMPLE database. At KANAGA, a primary database named SAMPLE is created. A standby HADR database is also created at the ATLANTIC host with port 50030.

At the client machine, the SAMPLE database is cataloged at the KANAGA node. The KANAGA node references the KANAGA host name and port 50030.

To activate ACR, you must update the alternate server for database SAMPLE at KANAGA by running the following command:

```
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 must run the following command on host ATLANTIC:

```
db2 update alternate server for database sample using hostname kanaga
port 50030
```

Without having the ACR feature set up, if there is a communication error, the application receives SQL error code -30081. When the primary database server fails, applications receive SQL error code -30081 even if HADR successfully brings up the standby server.

If the ACR 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 try only the failed transactions. The alternate server location is where the HADR standby is located. For client reroute to succeed, HADR must successfully start the standby server as the primary. This process must be initiated through the HADR **takeover** command either manually or by automating this takeover through some other HA software, such as IBM PowerHA SystemMirror for AIX.

10.4.3 ACR with a HADR database and PowerHA

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, the primary database SAMPLE is created. A standby HADR database is also created at host ATLANTIC with port 50030.

IBM PowerHA SystemMirror for AIX (PowerHA) detects when the active node is down and do the IP takeover and run the HADR **takeover** command to make the standby the primary server. In this case, PowerHA service IP address is configured as 9.43.86.111.

To activate ACR, you must update the alternate server for database SAMPLE at KANAGA using the service IP as the host name by running the following command:

```
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 must 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 there is a 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 environment with PowerHA, the service IP address (9.43.86.111) is used as the alternate server on both the primary and the standby servers. The service IP address (9.43.86.111) is also used as the host name when you catalog 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 PowerHA 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 the alternate server location (host 9.43.86.111 with port 50030). Assuming that PowerHA completes the IP takeover and moves 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 try the failed transactions.

In the HADR environment with PowerHA, we could also configure the alternate servers, as another option, similar to the process listed in 10.4.2, “ACR with a HADR database” on page 389.

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 the 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 the host name instead of the 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 PowerHA detects that the node is down and does an IP failover. When ACR detects a communication failure, it first tries to connect to the original server. There is a good chance that the PowerHA IP takeover and resource group transition 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.

PowerHA can make it all so seamless that sometimes when the client machine catalogued the database at a node using the Service IP address, it is not clear which host machine is the primary server. To discover the primary database server host name, run the following query:

```
db2 select substr(host_name,1,8) from table (db_partitions()) as t
```

10.5 Application programming to handle ACR

In this section, we show how to handle the ACR in application programming by using a Java application code sample.

10.5.1 ACR support for Java applications

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. 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 that is 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 the 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.

Domain Name System: In DB2, Domain Name System (DNS) is supported as a repository of alternate server information. For client reroute during connections to DB2 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 10.1.2, “ACR in action” on page 380.

For more information about ACR support for Java clients, see the following reference:

http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.lu.apdv.java.doc%2Fsrc%2Ftpc%2Fimjcc_c0056186.html

10.5.2 Implementing ACR on the DataSource interface with JDBC

In the following section, we use the *DataSource* interface with JDBC and explain how to set up the *javax.sql.DataSource* or *javax.sql.ConnectionPoolDataSource* interface with ACR.

Example 10-8 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 10-8 Obtaining a database connection using a `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();
```

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 by using the JNDI API. If, for example, 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 is updated, it is updated on the client machine without requiring your intervention. If the server is missing, the `.binding` file is read and a new connection attempt is made at the location of the alternate server. You can use LDAP to provide non-volatile storage for the alternate server information. You should not use volatile storage, as a client machine failure could result in the loss of alternate server data that is stored in memory.

To register the alternate server with JNDI and make it persistent, complete the following steps:

1. Create an instance of DB2ClientRerouteServerList, and bind that instance to the JNDI registry. DB2ClientRerouteServerList is a serializable bean with four properties:
 - alternateServerName
 - alternatePortNumber
 - primaryServerName
 - primaryPortNumber

Getter and setter methods for accessing these properties are provided.

2. Assign the JNDI name of the DB2ClientRerouteServerList object to DataSource property clientRerouteServerListJNDIName.

Example 10-9 shows the code sample of registering an alternate server with JNDI.

Example 10-9 Registering an alternate server with JNDI

```
// 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);

// Assign the JNDI name for the DB2ClientRerouteServerList object
datasource.setClientRerouteServerListJNDIName("serverList");
```

Here is how the IBM Data Server Driver for JDBC and SQLJ makes DB2ClientRerouteServerList persistent:

1. 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 to the client at connect time.

2. The IBM Data Server Driver for JDBC and SQLJ creates an instance of a referenceable object named 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 more 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.

3. 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.
4. The IBM Data Server Driver for JDBC and SQLJ attempts to propagate updated information to the JNDI store after a failover if the clientRerouteServerListJNDIName property is defined. If clientRerouteServerListJNDIName is specified, the primary server information that is specified in *DB2ClientRerouteServerList* is used for connection. If a primary server is not specified, serverName information that is specified on the DataSource is used.
5. 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.

For more information about how to register alternate server with JNDI, see:

http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=%2Fcom.ibm.db2.luw.apdv.java.doc%2Fsrc%2Ftpc%2Fimjcc_c0056193.html

10.5.3 ACR exception handling in Java applications

After a connection is re-established using ACR, 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.

All the work that occurred within the current transaction is rolled back. Thus, in the application, you must check the reason code that is returned with the error and run all SQL operations that occurred during the previous transaction.

Example 10-10 shows that the Java application code that is handling the SQL exception -4498.

Example 10-10 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 do not have to re-tried when we get client
reroute exception

do { // while loop start
try {
// transcation logic
pstmt.setInt(1,10);
int rowsUpdated = stmt.executeUpdate(
    "UPDATE employee SET firstnmre = '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)
```

Failover for ACR: Failover for ACR can also be seamless in DB2. In this case, the SQLException with error code -4498 is suppressed so that the IBM Data Server Driver for JDBC and SQLJ can indicate that a failed connection is re-established.

The following conditions must be satisfied for seamless failover to occur:

- ▶ The enableSeamlessFailover property is set to DB2BaseDataSource YES.
- ▶ The connection is not in a transaction. The failure must occur when the first SQL statement in the transaction is run.
- ▶ All global session data is closed or dropped.
- ▶ The application is not a stored procedure.
- ▶ There are no open and held cursors.
- ▶ Autocommit is not enabled. Seamless failover can occur when autocommit is enabled. However, the following situation can cause problems.

Suppose that SQL work is successfully run and committed at the data server, but the connection or server goes down before acknowledgment of the commit operation is sent back to the client. When the client re-establishes the connection, it replays the previously committed SQL statement. The result is that the SQL statement is run twice. To avoid this situation, turn autocommit off when you enable seamless failover.



HADR configuration parameters and registry variables

In this chapter, we describe the DB2 configuration parameters and registry variables that affect HADR. We also describe the important factors to consider when you implement a HADR solution to obtain optimum HADR performance.

This chapter covers the following topics:

- ▶ DB2 HADR configuration parameters
- ▶ DB2 HADR registry variables
- ▶ Considerations

11.1 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 you set up HADR. However, it is always useful to know what to change and why, if an architectural requirement is changed later.

We do not provide specific performance recommendations in this section, because each environment requires tuning to match its own unique characteristics. The information that is provided here, in the DB2 manual pages, and other references that are listed, are enough to give you a good idea about which settings can meet your requirements.

11.1.1 Basic configuration parameters

Details about the configuration parameters can be found at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0011761.html>

Configuration parameters

The following parameters are available:

- ▶ **AUTORESTART**

Consider setting this **db cfg parm** to OFF when you use automatic client reroute (ACR) 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, leave **AUTORESTART** set to ON, as the database automatically activates when the instance is started and the first application connection attempt is made. The database attempts log/crash recovery as required for any inflight/indoubt transactions that exist at the time the database is deactivated.

When **AUTORESTART** is left OFF, any connection attempts receive an SQL1015N message.

- ▶ **LOGINDEXBUILD**

This **db cfg parm** should be set to ON so that index creation, recreation, or reorganization on the primary are logged and replayed on the standby databases.

In some cases, the replication of index activity might not be required, for example:

- a. A large but seldom accessed index.
- b. Indexes on temporarily used tables, such as staging tables.
- c. Where active log space is not enough to support logging of index builds.

The LOG INDEX build table attribute can be used to override the database level setting of **LOGINDEXBUILD** on individual tables.

► **LOGARCHMETHn**

Circular logging is not supported.

LOGARCHMETHn (1,2) specifies the destination for the primary and mirror archive logs. This destination can be a disk, IBM Tivoli Storage Manager, userexit, or a vendor supplied destination driver. Having a value here means that after an active log is closed by DB2 (no more active transactions refer to it), it is moved out of the active log directory and sent to the specified target/destination.

The **LOGRETAIN** parameter is replaced by **LOGARCHMETH1**. For more information, see the information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.wn.doc/doc/i0058741.html>

► **HADR_DB_ROLE**

This **db cfg parm** is configurable indirectly through HADR commands, such as **STOP**, **START**, and **TAKEOVER HADR**. The possible values are:

– STANDARD

The database is not in a HADR role and can be processed normally.

– PRIMARY

The database is the HADR primary, and all client interactions with the HADR pair occur here.

– STANDBY

The database is the HADR standby, and any client interaction attempts receive an SQL1776N message. Only the DB2 log replay engine is allowed to run against this database. You can run the following command against a 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 usage of an HADR-specific network card, which is addressed by its own IP address or host name, which is different from the “usual” name of the host.

▶ **HADR_LOCAL_SVC**

Logically coupled with the **HADR_LOCAL_HOST** parameter, this **db cfg parm** specifies the port number or service name that is used for the local HADR component of the HADR database. This parameter is separate from the port/service assigned to a DB2 Instance. The HADR service/port number is specific to each HADR database.

▶ **HADR_PEER_WINDOW**

When you set **HADR_PEER_WINDOW** to a non-zero time value, the HADR primary and standby database pair continues to behave as though they are in a Peer state, for the configured amount of time, if the primary database loses connection to the standby database. This parameter ensures data consistency.

▶ **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. There is no equivalent **db cfg parm** value for the local server; it can be derived from the current instance environment variable.

▶ **HADR_REMOTE_SVC**

Combined with **HADR_REMOTE_HOST** and **HADR_REMOTE_INST**, this **db cfg parm** specifies the port number or service name that is used by the HADR component for the HADR database on the remote server. As with **HADR_LOCAL_SVC**, this value is specific to the database, and any pair of HADR databases must have two different TCP/IP port numbers.

▶ **HADR_SYNCMODE**

This **db cfg parm** is important. The setting must match on both servers in the HADR pair. The following values are possible:

– **SYNC**

In this mode, log writes are successful only when logs are written to log files on the primary database and when the primary database receives acknowledgement from the standby database that the logs are written to log files on the standby database. The log data is stored on both servers.

- NEARSYNC

In this mode, log writes are successful only when the log records are written to the log files on the primary database and when the primary database receives acknowledgement from the standby system that the logs are written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site is not transferred to nonvolatile storage with all of the log data that it received.

- ASYNC

In this mode, log writes are considered successful only when the log records are written to the log files on the primary database and are 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.

- SUPERASYNC

In this mode, log writes are considered successful when the log records are written to the log files on the primary database. Because the primary database does not wait for log send to the standby database, transactions are considered committed regardless of the state of the replication of the transaction. Because the transaction commit operations on the primary are not affected by the relative slowness of the HADR network or the standby server, the log gap between the primary and standby might increase.

- ▶ **HADR_TIMEOUT**

This **db cfg parm** specifies the time in seconds that the **DB2 HADR EDU** waits for any response from its HADR partner before it considers communication to have failed and closing the connection. If HADR was in a Peer state before it closes the connection, the primary would no longer follow Peer state semantics when the HADR connection is closed. The response could be either heartbeat or acknowledgement (ACK) signals. There is no timeout for an 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 transactions after the HADR connect state changes to disconnect. This parameter value must be the same between the primary and standby databases. If **HADR_SYNCMODE** is set to ASYNC or SUPERASYNC, or **HADR_PEER_WINDOW** is set to 0, DB2 ignores this parameter. When you close the HADR connection by running **deactivate database**, 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 attempts to rebuild invalid indexes, and specifically for HADR, whether this action occurs 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, overrides that value.

Possible values are:

- SYSTEM

Applies to **db cfg** only; accept the value for **INDEXREC** in the **dbm cfg**.

- ACCESS

Rebuilds an invalid index when the index is first accessed, and then rebuilds on HADR standby during log replay.

- ACCESS_NO_REDO

Rebuilds an invalid index when an index is first accessed, but leaves the invalid index on the HADR standby (there is no rebuild during log replay on the HADR standby). The index is rebuilt after a HADR takeover and the first access of the underlying table.

- RESTART

This value is the value. A rebuild of an 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 run 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

A rebuild of an invalid index occurs after a **RESTART DATABASE** on the normal or primary database, but not on the standby during HADR replay. A rebuild occurs only after HADR takeover.

► **SELF_TUNING_MEM**

This parameter determines whether the memory tuner dynamically distributes available memory resources as required between memory consumers that are enabled for self-tuning. The memory consumers that can be enabled for self-tuning include 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 be active only on a HADR primary database.

▶ **LOGFILSZ**

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 **LOGFILSZ** is ignored). This value is kept even after a HADR role switch or takeover until the database is restarted, when the local **LOGFILSZ** value is used for the active log file size.

11.1.2 Automatic client reroute configuration parameters

Strictly speaking, this subset of DB2 configuration parameters is not part of HADR, or even a configuration parameter, but it is used by ACR.

The mechanism of ACR, and how it is a separate entity from HADR, is explained in Chapter 10, “Automatic client reroute” on page 377.

11.2 DB2 HADR registry variables

In this section, we examine the DB2 registry and operating system shell environment variables that pertain to HADR.

▶ **DB2_HADR_BUF_SIZE**

This registry variable is recognized only 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 eventually become “full”, preventing the standby from receiving more logs. Saturation of the receive buffer causes transactions on the primary to be blocked until the receive buffer has more room to receive log pages.

If the synchronization mode is ASYNC, the network pipeline from the primary to the standby eventually becomes full and the primary may not send any more logs. This situation is called *congestion*. In ASYNC mode, network congestion and saturation of the standby's receive buffer stall the primary log processing. Hence, transactions on the primary are blocked until congestion is cleared, provided the standby can send a 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 it sends logs, it still must wait for acknowledgement messages when the standby buffer is full. Thus, the primary transaction processing is eventually blocked in all the synchronization modes when the standby receive buffer is full.

A larger standby receive buffer size (set by the registry variable **DB2_HADR_BUF_SIZE**) allows 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 can still fill up and the primary is 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 benefit from a generous increase of the **DB2_HADR_BUF_SIZE** variable, while the DB2 product documentation might give the impression that only ASYNC mode benefits.

In SUPERASYNC mode, because the transaction commit operations on the primary database are not affected by the relative slowness of the HADR network or the standby HADR server, the log gap between the primary database and the standby database might continue to increase. It is important to monitor the log gap, as it is an indirect measure of the potential number of transactions that might be lost should a true disaster occur on the primary system. In disaster recovery scenarios, any transactions that are committed during the log gap would not be available to the standby database. Therefore, monitor the log gap by using the **HADR_LOG_GAP** monitor element; if the log gap is not acceptable, investigate the network interruptions or the relative speed of the standby HADR server and take corrective measures to reduce the log gap.

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 provides a solution that 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 a 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. The parameter has two possible values: NONRECOVERABLE and COPY YES.

NONRECOVERABLE (the default) leads to the following behavior. When a **db2 load...copy no** command is run on a HADR primary database, DB2 converts the standby to NONRECOVERABLE. The table space is placed in a copy pending (read only) state, and the HADR standby ceases to match the primary. The table on the standby is marked bad, and logs from the primary are not applied there. This state can be corrected by running **db2 load...copy yes**.

COPY YES: If this registry variable is set to COPY YES (with a valid copy destination as part of the syntax), the load is converted so that it automatically is used with log replay on the HADR standby to maintain data integrity. The **load** command still must have a valid target that the HADR standby database can access in order for this scenario to be effective (that is, a shared drive with matching path, or a mutually accessible Tivoli Storage Manager 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 situation can happen when shared disk relative mapping does not match on both servers, or a method of file transfer outside DB2 control occurs. Essentially, the value of **DB2LOADREC** is set to a plain text file that contains all required media information about the load copy backup image. The structure and required content of this file is fully described in the Information Center at:

<http://publib.boulder.ibm.com/infocenter/db21uw/v10r1/topic/com.ibm.db2.1uw.admin.dm.doc/doc/c0004595.html>

As of DB2 10, there are no operating system shell environment variables that are used to influence the operation of HADR.

11.3 Considerations

In this section, we point out important things to consider when you implement a HADR solution. There is no one way to configure HADR; everything depends on your 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 transaction performance. You must weigh the pros and cons of each of the following factors on the user requirements when you tune the HADR.

11.3.1 DB2 transaction performance

A HADR setup could have a slight impact on the performance of the DB2 transactions. With a stricter log shipping mode, the potential performance impact is higher. The tuning in a HADR setup is like the usual tuning of DB2 parameters in a non-HADR environment. There are no specific HADR parameters to be tuned. Each environment requires tuning to match its own unique characteristics.

Depending on the log shipping mode, the performance of the in-flight transactions on the primary can be 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 detects 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 the primary database can detect the failure and close the connection quickly. Closing the connection causes the primary database to change from a Peer state to a Disconnected state, which unblocks pending transactions. The HADR state changes from Peer to disconnected.

If the primary machine is unable to report the standby or network failure in time, DB2 relies on **HADR_TIMEOUT** to disconnect. If the primary does not receive a message from the standby for the number of seconds specified in **HADR_TIMEOUT** (the default is 120 seconds) since the last heartbeat, it changes the state to disconnected.

With the SYNC and NEAR SYNC log shipping modes, a commit statement for in-flight transactions does not complete until the HADR state is changed to disconnected. This situation occurs because for SYNC and NEARSYNC modes, the primary must wait for the acknowledgement. With ASYNC log shipping mode, in-flight transactions do not suffer from a network disconnection or from the standby being down, and do not need a commit statement for them complete immediately. In ASYNC mode, when the primary sends logs, the operating system network layer might return success, but buffers the data in the network path, or returns a congestion error.

For ASYNC mode, if the primary keeps sending logs, it eventually causes congestion. If a send returns congestion, the primary transaction processing is blocked immediately in ASYNC mode. When there is 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 that the standby does not drop the connection after the **HADR_TIMEOUT** value is exceeded.

For SUPERASYNCH mode, log writes are considered successful as soon as the log records are written to the log files on the primary database. Because the transaction commit operations on the primary are not affected by the failure of the standby server or the network, the primary is unaffected by the outage. The log gap between the primary and standby increases while the problem persists. The longer the outage, the higher number of transactions that might be lost if a disaster occurs on the primary system.

On the primary side, because the primary is receiving heartbeat from the standby, the primary does not drop the connection after it reaches the threshold for **HADR_TIMEOUT**. Because the standby can send heartbeat and the primary can receive the standby's heartbeat, the HADR pair stay connected. In the worst case, an unresponsive standby or network could cause the primary transactions to be blocked while the primary can receive heartbeat from the standby.

11.3.2 How to reduce takeover time

You want to keep HADR takeover time to a minimum. By reducing the amount of transaction rollback after takeover on the standby, you can access the database sooner.

There is not a significant difference in takeover time between the different log shipping modes. When a 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 a Peer state.

If you know with certainty that the primary failed and wants to failover, issue a forced takeover on the standby. If the standby has not detected the failure and is still in a Peer state, you can use unforced takeover. But because the primary is down, unforced takeover does not receive a response from the primary and hangs for the **HADR_TIMEOUT** period. So, if you determine that the primary is down, you should issue forced takeover directly.

Set **LOGINDEXBUILD** to ON. If the index changes are logged on the primary, propagated to the secondary, and applied there, they do not need to be rebuilt in the case of a takeover (for more information, see 11.3.10, “Index logging” on page 427).

11.3.3 Seamless takeover

HADR does not automatically detect a failed primary and issue a takeover; this process is manual. When you determine that the primary is down, you can run **takeover**. But manual takeover might not be acceptable in certain situations. In that case, you can configure HADR with a clustering software such as IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP), PowerHA, VCS, or MSCS to make this takeover seamless. The cluster software detects that the primary is down and issues a HADR takeover. For more information about implementing this solution, see Chapter 7, “HADR with clustering software” on page 223.

11.3.4 Performance implications of HADR_TIMEOUT

If one database does not receive a message from the other one at the end of the **HADR_TIMEOUT** period, the connection is closed.

Try to avoid setting the **HADR_TIMEOUT** threshold too high. In the Peer state, if the standby is not responding, transactions on the primary can hang for the **HADR_TIMEOUT** period. If a peer window is enabled (meaning that the **HADR_PEER_WINDOW** parameter is set to a non-zero value), each disconnection also leaves the primary in a disconnected Peer state for the configured peer window size duration, leaving transactions blocked on the primary.

Setting the **HADR_TIMEOUT** threshold too low can also impact 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 the Peer state. Set it to at least 60 seconds.

11.3.5 Applications with a high logging rate

For applications that have a high logging rate, you must ensure that the network can transmit 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. Use identical primary and standby hardware.

DB2 HADR does not currently support compression of the log files before you send them to the standby. However, depending on the synchronization mode that you are using, the logging rate on the primary can be reduced to the rate of the transfer in the network by choosing a more strict form of log shipping mode, such as SYNC mode. This stricter mode forces the primary to wait for the standby to acknowledge that it received the log record before it can proceed. The network bandwidth is the limit of the 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.

11.3.6 Network considerations

The primary database ships the log records to the standby database server over the Internet Protocol network when the primary does a log flush. In the primary database, SQL agent EDUs produce logs and write the logs in to the database log buffer, whose size is controlled by the database configuration parameter **LOGBUFSZ**. The **db2loggw** 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 generates a flush request. If there is no request, **db2loggw** still flushes the log buffer periodically. The size of each log flush is non-deterministic. When there are multiple client sessions, multiple requests can be combined. The logger is 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 causes a log flush. If the synchronization mode is SYNC or NEARSYNC and the network is not fast, the round-trip messaging that is 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, use a high speed, high capacity network between the primary and standby database. On such a network, the standby can receive and acknowledge the logs as quickly as possible. Also, ensure that the bandwidth of the network link is greater than the bandwidth of the logs that are generated *at peak times*. The network is vital to the performance of HADR, so it is important to tune the network. For better performance in a 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 sends 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 running **no -o tcp_recvspace=[value]**. You can also set interface-specific values of **tcp_recvspace** from the **smit chinet** 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 specified by the **tcp_sendspace** tunable value. You should set the **tcp_sendspace** tunable value to 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.

Other TCP parameters that can be tuned include:

- ▶ **rfc1323**
- ▶ MTU path discovery
- ▶ **tcp_nodelayack**
- ▶ **sb_max**
- ▶ Adapter options, such as checksum offload and TCP Large Send

For more information about TCP tuning parameters, see:

- ▶ http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/tcp_streaming_workload_tuning.htm
- ▶ http://pic.dhe.ibm.com/infocenter/aix/v7r1/topic/com.ibm.aix.prftungd/doc/prftungd/tcp_streaming_workload_tuning.htm

11.3.7 Network performance tips

In this section, we focus on network-related settings. You should consider the following items:

- ▶ Use a dedicated network for all HADR connections.
- ▶ Consider using multiple network adapters.

Network delays

Network latency affects transaction performance in SYNC and NEARSYNC modes. The slowdown in system performance as a result of using SYNC mode can be larger than the slowdown of the other synchronization modes. In SYNC mode, the primary database sends log pages to the standby database only after the log pages are successfully written to the primary database log disk.

To protect the integrity of the system, the primary database waits for an acknowledgement from the standby before it notifies 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 impact 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 impact to the primary database is minimal. The acknowledgement might have 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. The performance impact is even smaller with ASYNC mode than with NEARSYNC mode.

For SUPERASYNCH mode, log writes are considered successful as soon as the log records are written to the log files on the primary database. Again, network delay is not an issue.

Network down

The primary database is continuously polling the network. If the network goes down, the primary database detects the condition as soon as the primary host machine operating system detects the condition. If the primary machine and the primary database can detect the failure, it closes the connection quickly. Closing the connection causes the primary database to change from the Peer state to the disconnected state, which unblocks pending transactions.

If the primary machine is unable to report a network failure in a timely manner, DB2 relies on the **HADR_TIMEOUT** parameter to perform a disconnect. If the primary does not receive a message from the standby for the time that is specified in the **HADR_TIMEOUT** parameter, the primary disconnects. 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, the logs 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 eventually causes 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 that the standby does not drop connection after the **HADR_TIMEOUT** threshold is reached. On the primary side, because the primary is receiving heartbeat from the standby, the primary does not drop the connection either after the **HADR_TIMEOUT** threshold is reached. Because the standby can send heartbeat and the primary can receive the standby's heartbeat, the HADR pair stay connected. In the worst case, an unresponsive network causes primary transactions to be blocked until the primary does not receive any heartbeat or messages from the standby for the time that is specified in **HADR_TIMEOUT**. If no heartbeat is received by the primary from the standby until the **HADR_TIMEOUT** threshold is reached, 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 run **TAKEOVER BY FORCE** when there is a data gap between the primary database and the standby database. The primary continues to listen on the HADR TCP/IP port that is waiting for the standby to come online. When the network recovers, the standby database tries to catch up to the primary until both databases return to the Peer state.

Figure 11-1 shows what happens when the network is down between the primary and secondary.

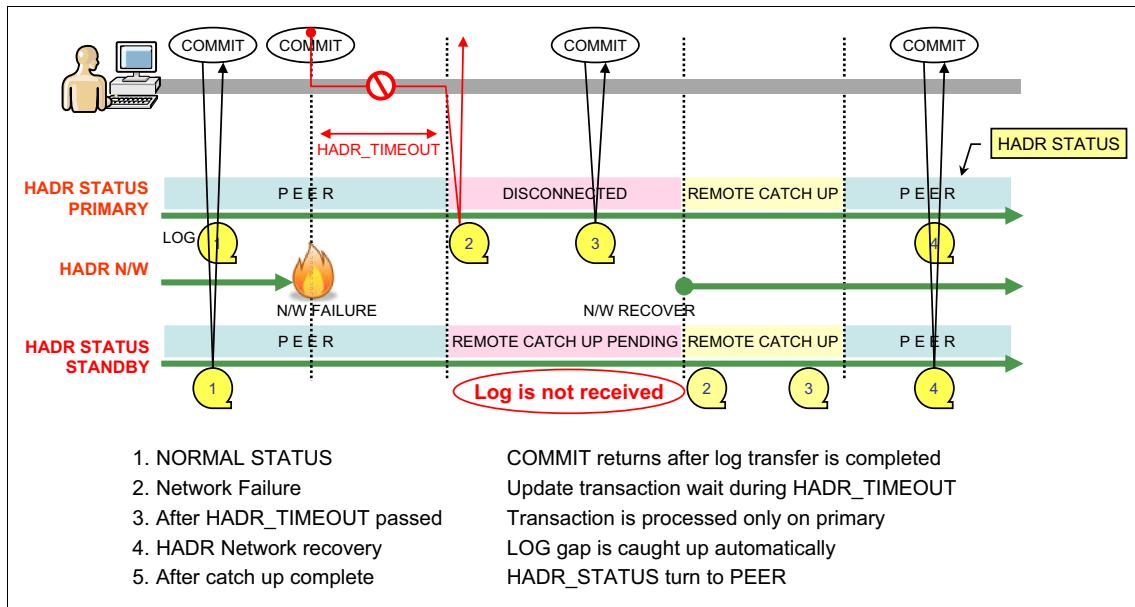


Figure 11-1 Behavior of HADR when the primary cannot transfer logs to the standby

Network congestion

Network congestion can happen when the standby is behind in receiving and processing log data from the primary. This delay can cause the standby's log receive buffer to fill up, preventing the buffer from receiving more log pages. This situation causes 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, which consists 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 the data, it cannot acknowledge the data. So the primary is blocked because it is waiting for the acknowledgement. Congestion is not likely to occur for SYNC and NEARSYNC modes while this log data is acknowledged, and no more logs are sent by the blocked primary.

In ASYNC mode, the primary 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.

In SUPERASYNC mode, log writes are considered successful as soon as the log records are written to the log files on the primary database. Because the primary database does not wait for log send to the standby database, network congestion has no effect on the primary. Slowness of the HADR network can cause the log gap between the primary and standby to increase.

Large jobs, such as table reorganization, can flood the standby and cause congestion, because 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 Internet Protocol network to increase Internet Protocol network buffering and also increase the value of the **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. Running **snapshot** or **db2pd** reports the connection status as congested. Congestion is reported by the **hadr_connect_status** monitor element.

11.3.8 Avoiding transaction loss in a HADR with HA cluster software

A HADR with an HA cluster software implementation automates the HADR takeover process. If a network error is detected, the primary database enters the disconnected state and lets the transactions proceed. This behavior maximizes the availability of the HADR database. Although the transactions can still be committed while the database is in a disconnected state, these transactions do not reach the standby and exist only on the primary database.

If the primary database goes down before the network recovers, the automatic takeover process starts. Because the primary database is down, the standby takes over the primary by force. The log gaps are not handled and the log records that are not transferred to the standby database are lost on the new primary database.

Figure 11-2 shows the behavior of a HADR database when the primary database cannot transfer logs to the standby database.

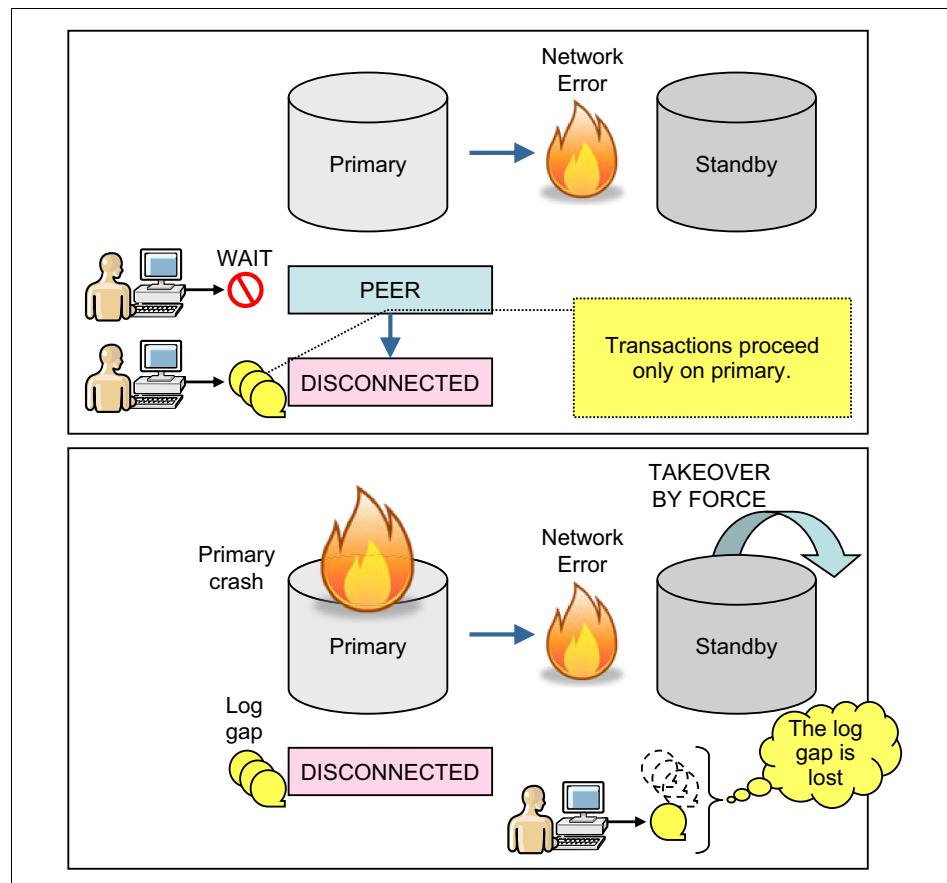


Figure 11-2 HADR behavior when the primary cannot transfer logs to the standby

Reducing network errors: To reduce the possibility of network errors, have duplicate networks for HADR communication.

To avoid data loss, the automated failover should be performed only if the HADR is in the Peer state when the primary crashes. Before you execute HADR takeover, ensure that the HADR is in the Peer state, which means that there is no log gap between databases in SYNC or NSYNC mode. ASYNC mode is not described here, because this mode does not guarantee the integrity of databases by its characteristics.

By monitoring the HADR status correctly and adding handling logic in an PowerHA clustered environment, you can ensure that the committed data is not lost after the standby system takes over the database. Figure 11-3 illustrates the concept of saving the committed data after the standby takeover.

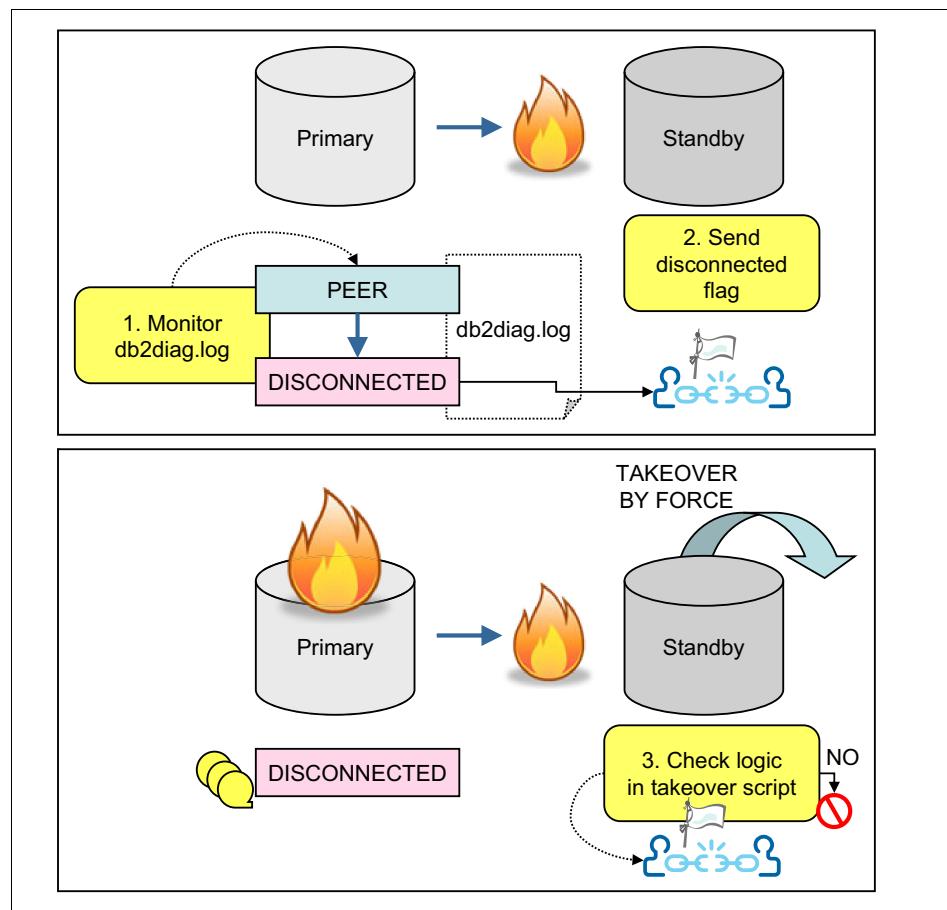


Figure 11-3 Avoiding data loss in a HADR with HA cluster software

As shown in Figure 11-3, the following steps are completed:

1. Monitor diag.log on the primary database:

You must detect the change of HADR status immediately when it turns into the disconnected status. The best way to do this task is by monitoring db2diag.log messages on the primary database.

Example 11-1 shows a message in db2diag.log when the state changes 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 11-1 HADR state that is changed from Peer to RemoteCatchupPending

```
2012-08-04-22.06.01.964755-240 E14817E379           LEVEL: Event
PID      : 9634             TID  : 47103946516800PROC : db2sysc
INSTANCE: hadrininst NODE : 000
EDUID    : 27               EDUNAME: db2hadrp (SAMPLE)
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-RemoteCatchupPending (was P-Peer)
```

Example 11-2 shows a message in db2diag.log when the state changes from RemoteCatchupPending to Peer.

Example 11-2 HADR state changed from RemoteCatchupPending to Peer

```
2012-08-05-22.06.03.304770-240 E23429E378           LEVEL: Event
PID      : 9634             TID  : 47103946516800PROC : db2sysc
INSTANCE: hadrininst NODE : 000
EDUID    : 27               EDUNAME: db2hadrp (SAMPLE)
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-NearlyPeer (was P-RemoteCatchup)
```

```
2012-08-05-22.06.03.317330-240 E23808E369           LEVEL: Event
PID      : 9634             TID  : 47103946516800PROC : db2sysc
INSTANCE: hadrininst NODE : 000
EDUID    : 27               EDUNAME: db2hadrp (SAMPLE)
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-Peer (was P-NearlyPeer)
```

To keep tracking through db2diag.log, run **tail**. Example 11-3 shows a sample script (**tail_hadr_status.ksh**). When this monitoring script detects the changes of the HADR state by tracking the messages in db2diag.log, it writes the state to the file. For example, set “1” when the HADR state is changed from Peer to non-Peer, and set “0” when the HADR status is returned from non-Peer to Peer. This state flag file allows the standby system to know the HADR state when the primary is down.

Example 11-3 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 run this monitoring script 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 Appendix B, “IBM Tivoli System Automation for Multiplatforms takeover scripts” on page 541.

2. Notify the standby system about the HADR state.

Because the **takeover** command is run on the standby database, the standby system must know the HADR state of the primary database to act. This status flag should be accessible from the standby node even though the primary node crashed. In our example, the status flag is on the standby node and updated from the primary node by running a remote command. Especially in an PowerHA cluster, **c1_nodecmd** is useful because this command can use any available network that is defined in the PowerHA cluster, which means **c1_nodecmd** can be run over one of the available networks even if a network interface for HADR is down. PowerHA is configured with multiple network cards in many cases. For more details, see Chapter 2, “DB2 with IBM Tivoli System Automation for Multiplatforms” on page 19.

Example 11-4 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`). The parameter value `$1` (0 or 1) is provided by `tail_hadr_status.ksh`, and `c1_nodecmd` writes it in the flag file on the standby node.

Example 11-4 write_status.ksh

.....

```
usr/es/sbin/cluster/sbin/c1_nodecmd -cspoc '-n ${REMOTENODE}" "echo  
$1 > $STATUS_FLAG"
```

.....

3. Add integrity check logic to the takeover script:

In the takeover script, you must add the logic to check the status flag before you run **TAKEOVER HADR**. It should fail over only if the primary database is in the Peer state before it crashes. If the primary database is not in the Peer state before the moment it crashes, it does not run takeover automatically and notifies that user intervention is required only to bring the database online. You 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 11-4 on page 422 shows a summary of the events that occur and the actions that are taken to prevent data loss:

1. A 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 proceed only on the primary database, and are not transferred to the standby.
4. The HADR network recovers from the error and log catch up is automatically run.
5. After the catch up completes, the HADR status returns to a Peer state. The monitor script catches the message in `db2diag.log` on the primary database and sends the flag file to the standby node to notify it that the HADR status is in the Peer state.

- Meanwhile, the flag is set to non-Peer status “1” in event 1 and keeps the same status until event 5. You do not want to run takeover by force because there is the possibility of data loss. You can add the logic in the takeover scripts to check this flag before you run **TAKEOVER HADR** with the **FORCE** option.

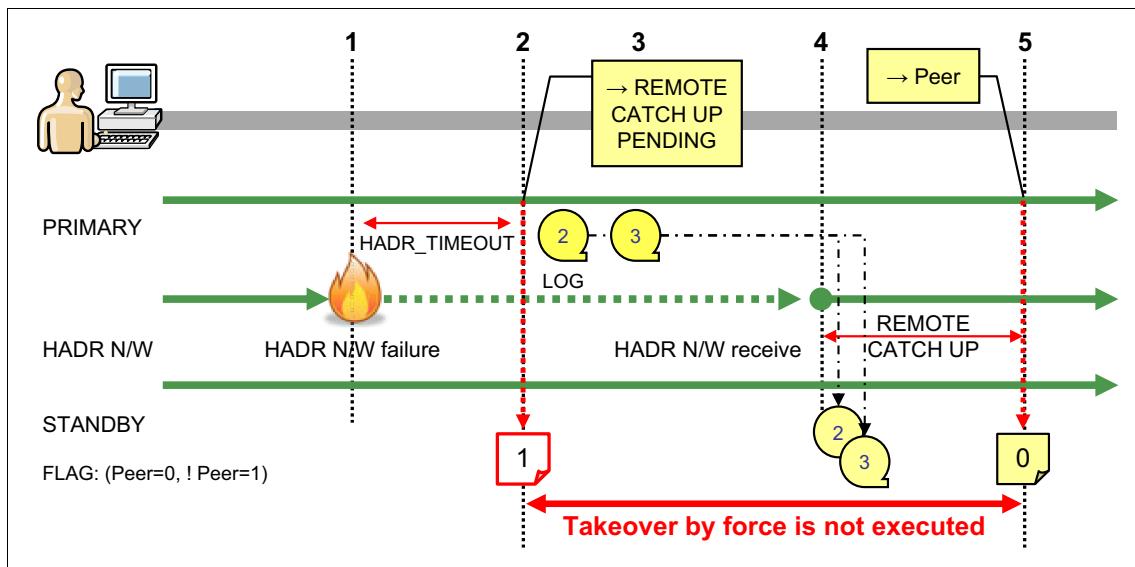


Figure 11-4 The behavior of the implementation

11.3.9 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, adopt the carefully designed architecture that is described in 11.3.8, “Avoiding transaction loss in a HADR with HA cluster software” on page 416. Now you can use the peer window to avoid this issue. In this section, we describe the peer window and show the typical scenarios of HADR takeover without risking data loss.

The DisconnectedPeer state

The HADR peer window introduces a new HADR state: **DisconnectedPeer**. If the HADR peer window is enabled, the HADR state is not changed from the **Peer** state to the **RemoteCatchPending** state directly. Instead, the HADR state is changed to **DisconnectedPeer** first. This period is the *peer window phase*. In this phase, the primary database does not accept the commit requests. The standby database can be switched to the primary database without the risk of data loss in this peer window phase.

Figure 11-5 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, you can perform HADR takeover without data loss.

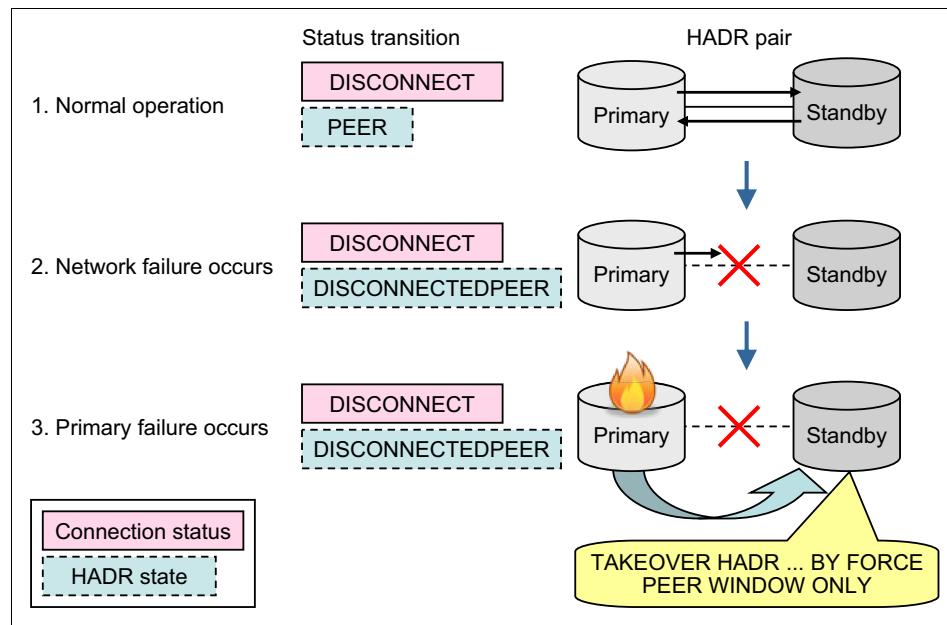


Figure 11-5 The typical behavior of HADR with the peer window

Takeover of HADR without the risk of data loss

To perform the takeover without data loss, you must ensure that the takeover is taking place only when there is no unsent data in the primary database. Use the **PEER WINDOW ONLY** keyword to achieve this task. The **TAKEOVER HADR** command with the **PEER WINDOW ONLY** keyword succeeds only when the HADR pair can take over without the risk of data loss.

The following two scenarios illustrate how the peer window works when you perform HADR takeover.

Scenario 1: A successful takeover in the peer window phase

Figure 11-6 illustrates the progress of a successful takeover within the peer window phase.

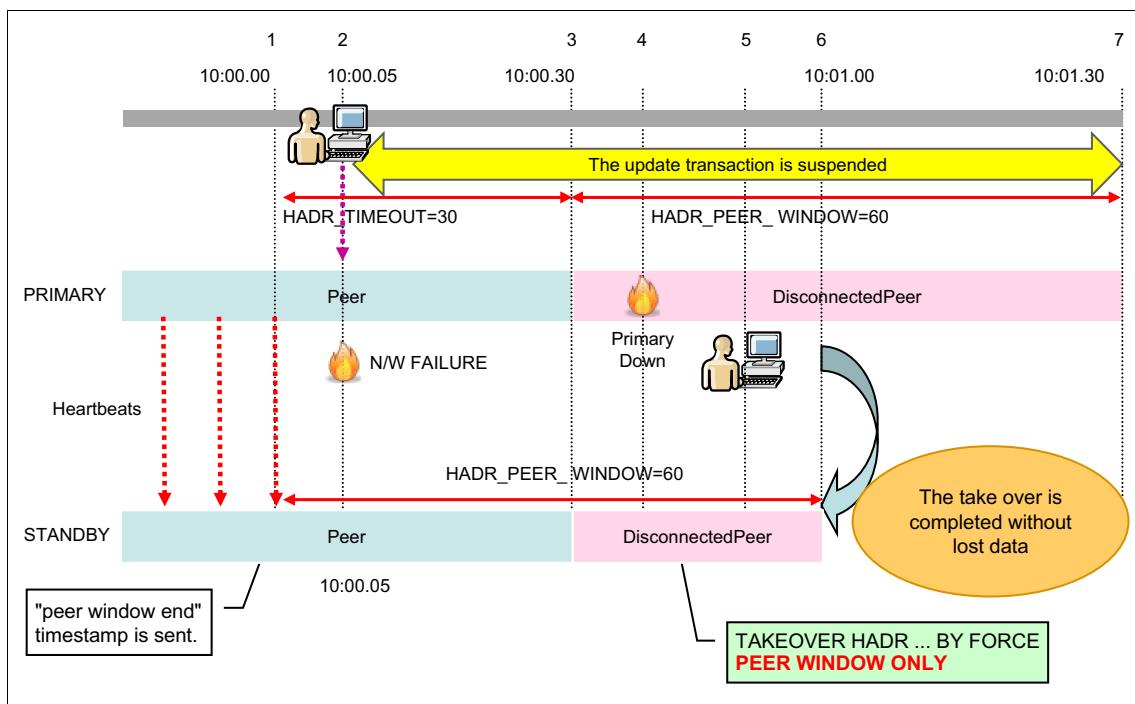


Figure 11-6 The successful takeover in the peer window phase

Where:

1. The HADR pair is operating normally and the HADR state is Peer. During the Peer state, the primary database sends the heartbeat with the PeerWindowEnd time stamp to the standby database regularly. In this example, the standby database receives the last heartbeat at 10:00:00 and recognizes that the PeerWindowEnd time stamp is 10:01:00.
2. A network failure occurs 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 running **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** on the standby database.

6. Complete the takeover by the time of PeerWindowEnd without the unsent transaction log occurs.
7. If the primary database did not fail, after the peer window phase on the primary database expires, the primary database starts to accept the update transactions from this point.

Scenario 2: A failed takeover outside of the peer window phase

Figure 11-7 illustrates a failed takeover scenario because of the expiration of the peer window phase.

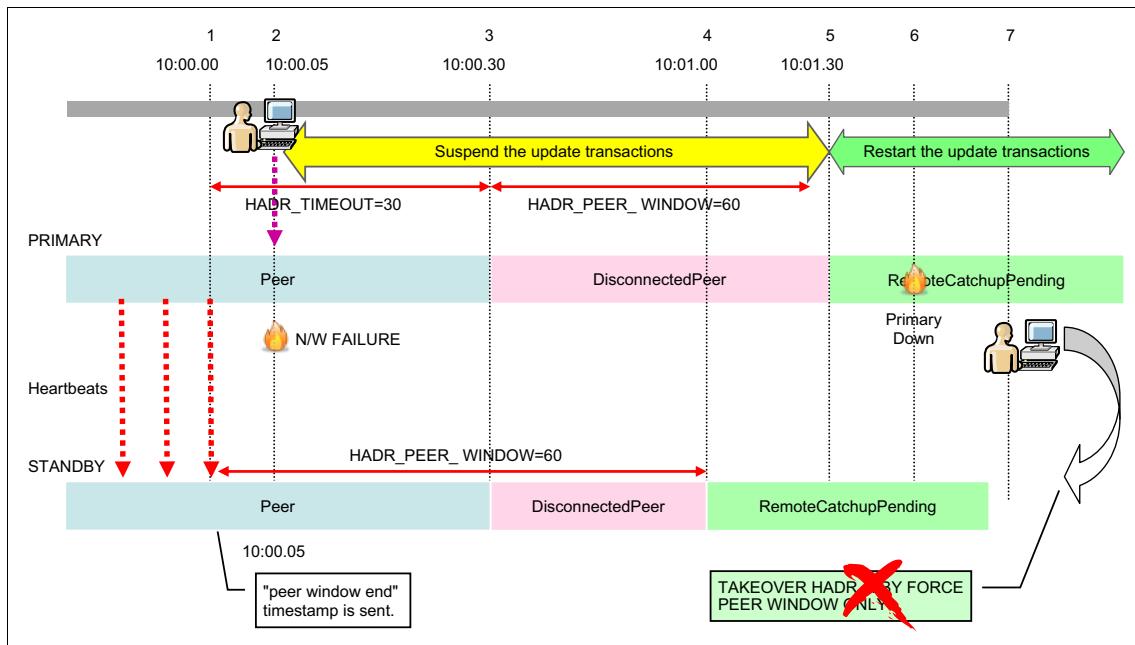


Figure 11-7 The failed takeover the outside of the peer window phase

Where:

1. The HADR pair is operating normally and the HADR state is Peer. During the Peer state, the primary database sends the heartbeat with the PeerWindowEnd time stamp 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 time stamp is 10:01:00.
2. A network failure occurs 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 starts to accept the update transactions because the peer window phase on the primary database expires.
6. The primary database fails.
7. Start the takeover by running **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY** on the standby database. The **TAKEOVER HADR** command fails because the peer window phase expired.

Considerations

There are a few considerations regarding the usage of the peer window:

- ▶ Both servers should have the same system clock.

A PeerWindowEnd time stamp is generated on the primary database that is 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 situation 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 run.
- ▶ 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 values of the database configuration parameters, or the application can time out faster than the database operations do. Therefore, when you use the peer window, you must incorporate the peer window time in to the application timeouts. When the primary database cannot send the transaction logs to the standby database, the update transactions are suspended at least the length of time specified in **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 specified in **HADR_TIMEOUT** and **HADR_PEER_WINDOW**.

- ▶ If the HADR is being used 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 run **TAKEOVER HADR ... BY FORCE PEER WINDOW ONLY**. The **takeover** HADR command with **PEER WINDOW ONLY** requires the **DisconnectedPeer** status. If the HADR status is changed to **RemoteCatchupPending** before you run **takeover**, this script fails with the message SQL1770N.
- ▶ If you run **db2haicu** to configure 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**, see Chapter 7, “HADR with clustering software” on page 223.
- ▶ If **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”.

11.3.10 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 situation is undesirable for HADR, because a standby has pending build indexes that are rebuilt when someone accesses them. On each takeover, you have indexes that are marked as bad, and 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 rule controls whether to log an index build or not. An index update is always logged. This parameter applies to index creation, recreation, and table reorganization. The default is OFF. Set 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 an index build.
- ▶ OFF: Does not log an index build.
- ▶ NULL: Defaults to what **LOGINDEXBUILD** is set to in the 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 that are rebuilt or reorganized, 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 reorganizing indexes could impact transaction performance.

The database manager and database level configuration parameter **INDEXREC** specifies when the invalid index is rebuilt. For more details about the values for this parameter, see 11.1, “DB2 HADR configuration parameters” on page 400.

11.3.11 Backup from standby image with FlashCopy

With the Storage Copy function, you can back up the HADR standby database without impacting the performance of the primary database.

To take a backup image from the standby database, complete the following steps:

1. Complete a database backup on the standby server.

Suppose that 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. Complete 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 FlashCopy, 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 it waits for the physical copy of whole storage area. With IBM Total Data Storage, after the logical copy is finished, you can activate the standby image again.
 - e. Activate (varyon) volume groups, mount file systems, and activate the database. When the standby database is deactivated, you might not have the latest standby database.
2. Restore to the primary server.

If you must restore the database from this backup to the primary server, complete 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.
 - 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 is made on the standby server, its HADR status is 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 running).
 3. Restore to the standby server.

If the standby database becomes unusable because of a hardware problem, the split backup can be restored to the original standby database. No special configuration is required. After the standby database files are restored, start HADR as the 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 the Peer state.

11.3.12 Replicating load data

When the 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 run on the primary database with the **NONRECOVERABLE** option, the command runs on the primary database and the table on the standby database is marked bad. The standby database skips future log records that pertain to this table. You can run **LOAD** with the **COPY YES** and **REPLACE** options to bring back the table, or you can drop the table to recover the space.

The following methods can be used for replicating load data to the standby database:

- ▶ Use a network file system (NFS) shared disk.

You can use an NFS to share the load copy image on both primary and standby nodes (Figure 11-8). On the standby node, be sure to mount the NFS shared directory to the same point that the primary can see.

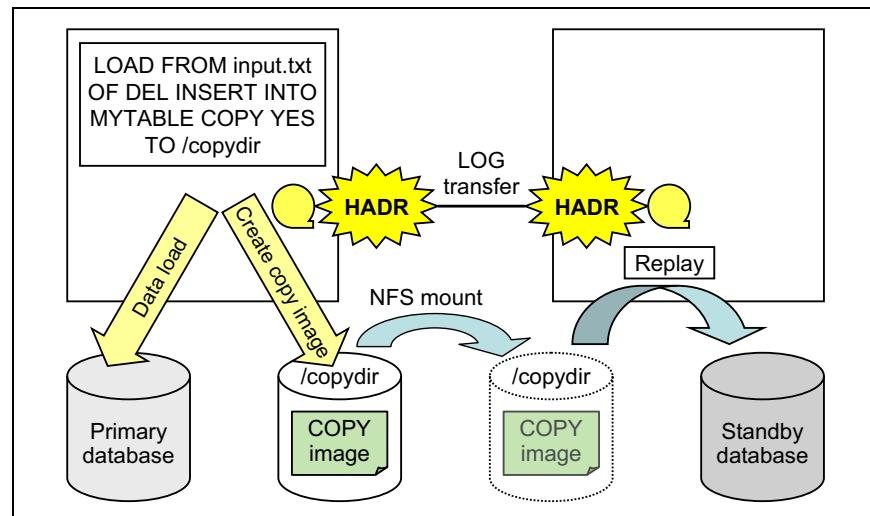


Figure 11-8 Using NFS to share data

- ▶ Pause (deactivate) the standby database while you transfer the load copy.
If you transfer the copy image by file copying or physical tape media, stop the standby (run `db2 deactivated db`) before the primary runs the load. After the primary finishes the load and the copy is in place on the standby, restart the standby (run `db2 activate db`). The standby reconnects and replays the load.

Ensure that instance owner has the correct access permission to the copy file. For example, in an AIX system, the permissions of copy image are set to 640 (rw-, r--, ---). The instance owner of the standby database should have same user ID (uid) or belong to the same group ID (GID) 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 must rebuild the standby database from the primary.

11.3.13 Log archive and HADR

Log archive happens only on the primary. If the primary and the standby use independent log archive devices with takeover, some log files are archived on one device and others on the other device. Log is never archived on the standby database even if you use **USEREXIT**, **LOGARCHMETH1**, or **LOGARCHMETH2**. The standby writes logs it receives from the primary in to its local log files. To avoid filling up the log path, it recycles log files after a file is no longer needed for crash recovery and takeover readiness.

11.3.14 Database restore considerations

Here 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 that are created by relative paths are restored to paths relative to the new database directory.
- ▶ Restoring the standby database has certain requirements.

When you create the standby database by restore from the primary database, the restore commands should leave the database in the rollforward pending state. Otherwise, starting the HADR standby fails with HADR error SQL1767N start HADR cannot complete. Reason code = "1". Do *not* specify the **without rollforward** when you rebuild the standby database.



Backup and recovery

In this chapter, we describe some advanced data recovery-related functions that are provided with DB2 for partitioned database environments.

Single System View (SSV) greatly simplifies the backup of multiple-partition databases. SSV also makes the process of recovery to a single stable point in time more robust and less error prone.

The snapshot backup provides a solution for many clients who require either 24x7 availability or have such large databases that a conventional backup is not practical.

Although DB2 provides a complete solution for recovery to a point-in-time or to end of logs, the **restore database** and **rollforward database** commands kept the process of recovery in two discrete steps. With the introduction of the **recover database** command, this situation changed. In addition, a valuable feature is added to the **rollforward database** command.

Throughout the evolution of DB2 versions on Linux, UNIX, and Windows platforms, there have been technical advances in how administrators can manage backups and archived transaction log files. For more information about these capabilities, see the topic “Data recovery” in the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/c0052073.html>

This chapter covers the following topics:

- ▶ Single system view backup
- ▶ Backup and restore database with snapshot backup
- ▶ Recover database command
- ▶ Recovery object management

12.1 Single system view backup

You can use the SSV backup to perform a multi-partitioned database backup similar to a non-partitioned database. Because clients with large data volume might use multi-partitioned databases, here is a brief introduction to this feature.

Previously, to take a database backup for a multi-partitioned database, you either ran **backup** on each partition one at a time or ran **db2_a11** to run the backup commands in parallel on all partitions. However, 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 time stamps regardless of how the database backup commands are performed. When a database restore is required, identifying all database partition backup images that have different backup time stamps for the same backup becomes complicated. Besides, the log files that are 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, cumbersome, and error-prone.

These difficulties are removed by the SSV backup. When you perform 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. Backups for the specified partitions are taken simultaneously, and the backup time stamp that is associated with all specified database partitions is the same. In addition, you can include database logs with an SSV backup. When you restore the database from an 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.

12.1.1 Using single system view backup

There is no installation or configuration that is required for using SSV backup. 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, into the image, all the log files required to recover the database to a point-in-time.

Performing the backup task

To create a database backup, you can run **backup** as usual on the catalog partition. The syntax for two simple SSV backup commands is shown here:

- ▶ **BACKUP DB sample ON ALL DBPARTITIONNUMS TO <backup directory>**
- ▶ **BACKUP DB sample ON DBPARTITIONNUMS 0 TO 3 TO <backup directory>**

The first command specifies the keyword **ON ALL DBPARTITIONNUMS**, which includes all database partitions for a SAMPLE database that is listed in the db2nodes.cfg file in the backup. The second one specifies the keyword **ON DBPARTITIONNUMS 0 TO 3**, which creates a backup for partitions with the database partition numbers 0 - 3.

When you want to include the log files in an SSV backup for minimum recovery, all you must do is to set the keyword **ONLINE**:

```
BACKUP DB sample ON ALL DBPARTITIONNUMS ONLINE TO <backup directory>
```

Specifying the **ONLINE** keyword in the **backup** command uses the **INCLUDE LOGS** option by default, unless it is a non-SSV backup on a partitioned database.

Example 12-1 shows that the SSV database backup for the multi-partitioned database SAMPLE is taken with a single command and all four database partition backup files have the same time stamps in the file name.

Example 12-1 An 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 :
20120717150711

```
$ ls /work/backup
total 681816
SAMPLE.0.db2inst3.DBPART000.20120717150711.001
SAMPLE.0.db2inst3.DBPART000.20120717150711.001
SAMPLE.0.db2inst3.DBPART000.20120717150711.001
SAMPLE.0.db2inst3.DBPART000.20120717150711.001
```

Checking the log file status

You can check if a database backup image includes the log files by running **db2ckbkp**. This command provides information about the specified database backup image file.

```
db2ckbkp -h <database backup file>
```

Example 12-2 shows the output of the **db2ckbkp -h** command for the previously created backup. The line field, Includes Logs, confirms if the backup image includes the log files or not. The value 1 means that this backup includes the log files.

Example 12-2 Sample output of the db2chbkp command

```
$ db2ckbkp -h SAMPLE.0.db2inst3.DBPART000.20120717150711.001

=====
MEDIA HEADER REACHED:
=====

Server Database Name      -- SAMPLE
    Server Database Alias      -- SAMPLE
    Client Database Alias      -- SAMPLE
    Timestamp                  -- 20120717658932
    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/SQL0GDIR/
```

Backup Buffer Size	-- 3280896
Number of Sessions	-- 1
Platform	-- 14...

Restoring the database and the LOGTARGET keyword

When you change from the default offline backup to online backup, the required log files are automatically included in the backup image to enable a minimum forward recovery. However, to use the log files for rollforward recovery, you must extract these files out of the database backup image (it is not done automatically). There are two ways to accomplish this task:

- ▶ Specify the **LOGTARGET** keyword in the **restore database** command. DB2 extracts the log files that are included in the backup image to the target directory specified by **LOGTRAGET** during the database restore process. The directory that is specified in **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.

The syntax for the relevant command is:

```
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 run. This command takes only the log files out of the backup image to the directory specified in the **LOGTARGET** keywords.

The target directory must be created.

The syntax for the relevant command is:

```
RESTORE DB <DB name> LOGS FROM <backup dir> TAKEN AT <backup
timestamp> ON <target db path> LOGTARGET <target log restore path>
```

For the complete syntax of the **restore database** command, see:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.cmd.doc/doc/r0001976.html>

Example 12-3 shows one way to restore a four partition database. The backup image is taken using SSV backup, as shown in Example 12-1 on page 435. In this example of a restore, we run **restore** with the **LOGTARGET** keyword on each partition to restore one partition at a time. All the database backup images carry the same time stamp in each restore on the four nodes. The directory for holding the log files is created manually on each physical node before restore starts.

Example 12-3 Example of a 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 20120717150711 on
/home/tukiv10/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 20120717150711
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 20120717150711
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 20120717150711
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 tukiv10 system          12288 July 17 13:25 S0000005.LOG

/work/logs/log1:
```

```
total 24
-rw----- 1 tukiv10 system 12288 July 17 13:25 S0000005.LOG

/work/logs/log2:
total 24
-rw----- 1 tukiv10 system 12288 July 17 13:25 S0000005.LOG

/work/logs/log3:
total 24
-rw----- 1 tukiv10 system 12288 July 17 13:25 S0000005.LOG
```

Another valuable example for rebuilding all database nodes in a partitioned environment can be found in the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/t0021537.html>

Tips: You also can run **db2_all** to run **restore** for all database partitions. The command syntax is as follows:

- ▶ For the **restore** command for the catalog partition, use:

```
$ db2_all "\<<+0< db2 restore db sample from /work/backup taken
at 20120717150711 on /home/tukiv10/db2 logtarget /work/logs/log## without prompting"
```

- ▶ For the **restore** command for the other partitions, run:

```
$ db2_all "\>||<-0< db2 restore db sample from /work/backup
taken at 20120717150711 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 must 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_all** command runs 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 quotation mark) 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

Individual backups for each partition cause the backup time stamps to be different for each backup image. To perform a forward recovery, you must identify the minimum (latest) recovery time stamp from all database partition backup images. SSV backup has one backup time stamp for all backup images.

Specify the backup time stamp 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 searches the log files in the log path, the archive log path, and the failover log path by default. If the log files are in a directory other than these default directories, specify the log file location in the **OVERFLOW LOG PATH** parameter. In a partitioned database environment, the *overflow log path* is the default overflow log path for all database partitions. You must specify one even if you do not use it. The overflow log paths for each database partition are listed after this path.

Example 12-4 is an example for **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 12-4 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  
  
Nodenumber  Rollforward    Next log      Log files processed   Last committed  
transaction  
           status        to be read  
-----  
-----  
2012-07-17-12.04.53.000000 UTC  0  DB pending    S0000005.LOG      -  
2012-07-17-12.04.51.000000 UTC  1  DB pending    S0000005.LOG      -
```

```

2 DB pending S0000005.LOG      -
2012-07-17-12.04.51.000000 UTC
3 DB pending S0000005.LOG      -
2012-07-17-12.04.52.000000 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 transaction	Rollforward status	Next log to be read	Log files processed	Last committed
0 2012-07-17-12.04.53.000000 UTC	not pending		S0000005.LOG-S0000006.LOG	
1 2012-07-17-12.04.51.000000 UTC	not pending		S0000005.LOG-S0000006.LOG	
2 2012-07-17-12.04.51.000000 UTC	not pending		S0000005.LOG-S0000006.LOG	
3 2012-07-17-12.04.52.000000 UTC	not pending		S0000005.LOG-S0000006.LOG	

DB20000I The ROLLFORWARD command completed successfully.

12.1.2 Considerations

Here are a few considerations for using SSV backup:

- ▶ The SSV **backup** command must be run on the catalog partition.

If you run the SSV **backup** command on two non-catalog partitions, the **backup** command might fail with an SQL4976N message.

- ▶ SSV backup is not enabled automatically.

Without specifying the keyword **ON DBPARTITIONNUM** or **ON ALL DBPARTITIONNUMS** in the **backup** command, the backup runs on the current database partition only.
- ▶ Only one backup task can be run in parallel on the database partition.

You must not run multiple database backup on one database partition. If you run the **backup** command on the non-catalog partition during the SSV backup, more **backup** commands might fail with an SQL1035N message.

12.2 Backup and restore database with snapshot backup

Many storage subsystems provide FlashCopy or snapshot functions, so you can make nearly instantaneous point-in-time copies of entire logical volumes or data sets.

DB2 provides integrated support for snapshot backup through DB2 Advanced Copy Services (ACS). You can use DB2 ACS to use the fast copying technology of a storage device to perform the data copying part of backup and restore operations. To perform snapshot backup and restore operations, you need a DB2 ACS API driver for your storage device. For a list of supported storage hardware for the integrated driver, see the Tivoli documentation at:

http://publib.boulder.ibm.com/infocenter/tsminfo/v6r3/topic/com.ibm.itsm.fcm.unx.doc/c_fcmu_ovr_supstorage.html

DB2 ACS is installed during a typical or custom DB2 installation and must be enabled before it is used. The steps to activate DB2 ACS are described at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.1uw.admin.ha.doc/doc/t0052799.html>

A backup operation that uses DB2 ACS is called a *snapshot backup*. To perform a snapshot backup or restore with DB2 ACS, you must specify **use snapshot** in the **backup** and **restore** commands.

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.

Complement snapshot backups with regular disk backups to be prepared for storage failure.

DB2 10.1 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**

To restore a database from a snapshot backup, run either **restore database** with the **use snapshot** parameter or the db2Restore API. DB2 10.1 does not support the **use snapshot** parameter in the **restore** command with any of the following parameters:

- ▶ **incremental**
- ▶ **to**
- ▶ **on**
- ▶ **dbpath on**
- ▶ **into**
- ▶ **newlogpath**
- ▶ **with num-buffers buffers**
- ▶ **buffer**
- ▶ **redirect**
- ▶ **replace history file**
- ▶ **compression library**
- ▶ **parallelism**
- ▶ **comprlib**
- ▶ **open num-sessions sessions**
- ▶ **history file**
- ▶ **logs**

If you do not specifically include log files in the restore, the **logtarget** parameter defaults to **exclude**.

12.3 Recover database command

The **recover database** command consolidates a number of steps that are previously required with the **restore database** and **rollforward database** commands.

While retaining some useful niche functionality, such as initializing an HADR standby database that must be left in the 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 **rollforward database** commands are unnecessary for common situations.

12.3.1 Feature summary

As a feature that is 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. Even with a recovery to point-in-time (PIT), DB2 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 time stamp and physical location to choose the correct backup file and the time stamp to roll forward to.

For deeper insight, the DB Information Center highlights the features of the **recover database** command at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.1uw.admin.cmd.doc/doc/r0011703.html>

The **RECOVER DATABASE** command has parameters that you can use to specify the following items:

- ▶ 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 run from the catalog node.
- ▶ Custom decompression libraries and options.
- ▶ A recover operation that follows a previously incomplete recover operation attempts to continue if possible. The **restart** option forces a new **restore** and **rollforward** operation. This action avoids lengthy waits for restart logic to find the appropriate place to commence.

Judging from the foregoing list of features, the **recover database** command is a balance between simplicity and versatility.

12.4 Recovery object management

A client environment might accumulate many recovery objects over time, such as backup images and logs of considerable size. Because storage space is not unlimited, unneeded objects must be identified and removed without damaging objects that are still required for a successful recovery.

DB2 provides two options to manage those recovery objects. The topic “Managing recovery objects” in the DB2 Information Center describes these options in detail at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/t0051365.html>



Q replication

Based on client requirements, data replication might be an option to implement high availability or disaster recovery environments. The IBM InfoSphere Replication Server provides two approaches for data replication:

- ▶ The first approach is called SQL replication because it is based on DB2 tables. Changes are captured in staging tables with SQL and then shipped to a target database and applied to the target table.
- ▶ The second approach is Q replication. It is based on a messaging infrastructure instead of staging tables. Changes are captured on the source and placed on message queues. This approach is described in this chapter.

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.

More information about replication capabilities can be found in the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.swg.im.iis.db.repl.intro.doc/topics/iiyrcintch100.html>

This chapter covers the following topics:

- ▶ Introduction to Q replication
- ▶ Unidirectional setup

13.1 Introduction to Q replication

Q replication is a replication solution that uses WebSphere MQ message queues to transmit changes from a source table to target tables.

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 take place from one source to one or many targets. In unidirectional replication, typically the target is used for read-only operations.

- ▶ Bidirectional

In *bidirectional Q replication*, tables on two servers replicate to each other. Changes that are made on either table are replicated to the corresponding table. You cannot replicate a subset of rows, and each table must have the same number of columns and data types for those columns, although 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 have 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 information about different Q replication configurations, see the DB2 Information Center at:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.swg.im.iis.prod.repl.nav.doc/topics/iiyrqcnccreatesubspubs.html>

Here is 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 that is called the *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, WebSphere 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*.

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 (Figure 13-1).

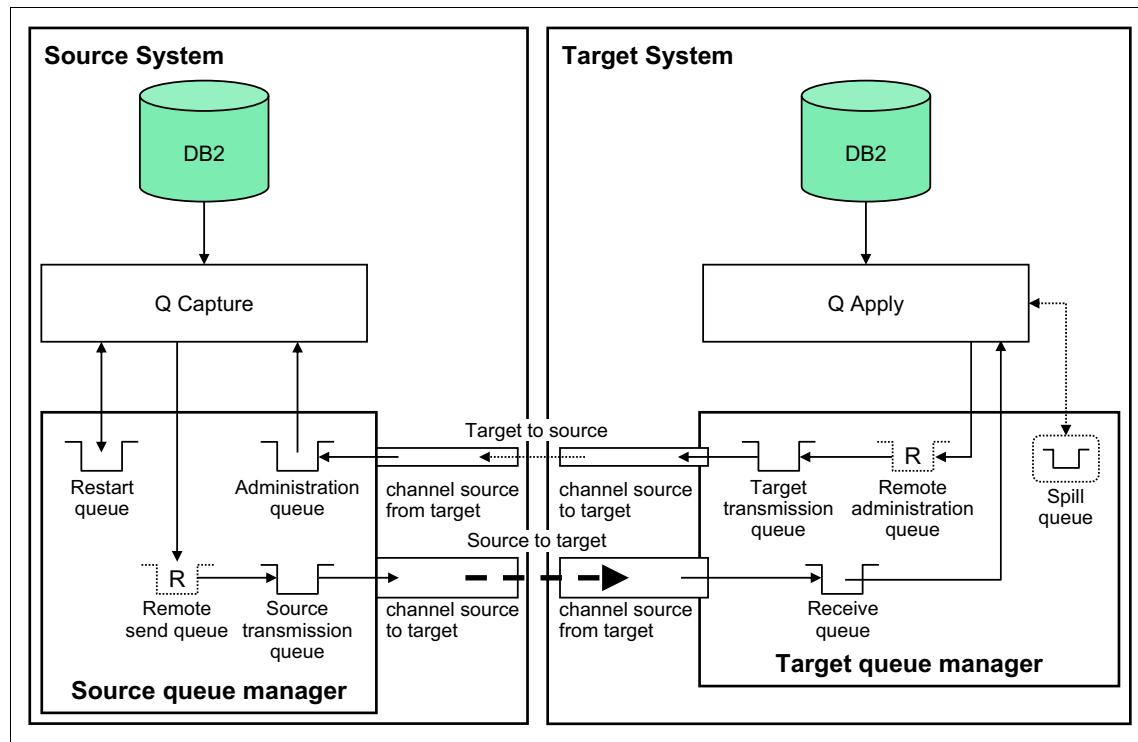


Figure 13-1 Unidirectional setup diagram

To set up unidirectional Q replication, complete the following tasks:

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 the target to source server connectivity.
9. Start Q Capture.
10. Start Q Apply.

In our example, we use two Windows servers that are running DB2 10.1 and WebSphere MQ V7.05. 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 logging method for the source must be changed from the default circular logging to archival logging. Hence, the source database (SAMPLE in our test case) must have the database configuration parameter **LOGARCHMETH1** set to LOGRETAIN as follows:

```
db2 update database configuration for sample using LOGARCHMETH1  
LOGRETAIN
```

A backup is required if you are setting **LOGARCHMETH1** to LOGRETAIN 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 by running the following command:

```
crtmqm QMGR1
```
 - b. Start the Q manager that you created by running the following command:

```
strmqm QMGR1
```

- c. To create the needed WebSphere 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 Creating WebSphere MQ objects on the source for QMGR1

```
**This script creates the following objects within queue manager  
QMGR1 **at the source:
```

```
**A local administration queue  
DEF QLOCAL('SAMPLE_ADMINQ') DESCRIPTOR('Capture Admin Queue') REPLACE  
  
**A local restart queue  
DEF QLOCAL('SAMPLE_RESTARTQ') DESCRIPTOR('Capture Restart Queue')  
REPLACE  
  
**A remote queue definition for Q Capture to put data messages  
DEF QREMOTE('SAMPLE2TGTDBQ') XMITQ('QMGR2')  
RNAME('SAMPLE2TGTDBQ') 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. QMGR2 is created in the next step below  
**Replace the ip with your ip and the port number with your port  
number  
DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(SDR) TRPTYPE(TCP)  
CONNNAME('nnn.nnn.nnn.nnn (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 than the source by running the following command:

```
crtmqm QMGR2
```
 - b. Start the Q manager that you created by running the following command:

```
strmqm QMGR2
```

- c. To create the needed WebSphere 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 Creating WebSphere MQ objects on the target for QMGR2

```
**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) +  
CONNNAME('nnn.nnn.nnn.nnn (1415') XMITQ('QMGR1') REPLACE  
  
**A receiver channel from QMGR1  
DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE
```

- d. Start both the listener and channel on the source and the target:
 - i. On the source, from a command window, start the listener by running the following command:

```
RUNMQSLR -T TCP -M QMGR1 -P 1415
```

- ii. On the target, from a command window, start the listener by running the following command:

```
RUNMQSLR -T TCP -M QMGR2 -P 1416
```

- iii. On the source, from a command window, start the channel:

To start the channel, first run **RUNMQSC** 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
```

- iv. On the target from a command window, start the channel by running the following command:

```
RUNMQSC QMGR2
```

```
START CHANNEL(QMGR2.TO.QMGR1)
```

```
END
```

4. (Optional) Test the queues:

Test the message queue setup by putting a message from the source system and trying to get the message from the target system. WebSphere MQ provides two programs for testing the messaging setup: **amqspput** and **amqspget**.

In the source in our example, we put a message, Test message to send, into the message queue (Example 13-3):

- a. Press Enter to send the message.
- b. Press Enter to end the prompt.

Example 13-3 Sending the test message from the source

```
>amqspput SAMPLE2TGTDQ QMGR1
Sample AMQSPUTO start
target queue is SAMPLE2TGTDQ
Test message to send
Sample AMQSPUTO end
```

- c. Example 13-4 shows how to receive the message from the target. Press Enter to end the prompt.

Example 13-4 Sending the test message from the target

```
>amqsget SAMPLE2TGTDBQ QMGR2
Sample AMQSGETO start
message <Test message to send>
no more messages
Sample AMQSGETO end
```

If the test message flow between source and target does not work correctly WebSphere MQ issues the ASN2288W error code.

5. Creating the Q Capture Control tables:

- a. From the Replication Center, click the **Replication Center** drop-down menu and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box (Figure 13-2).

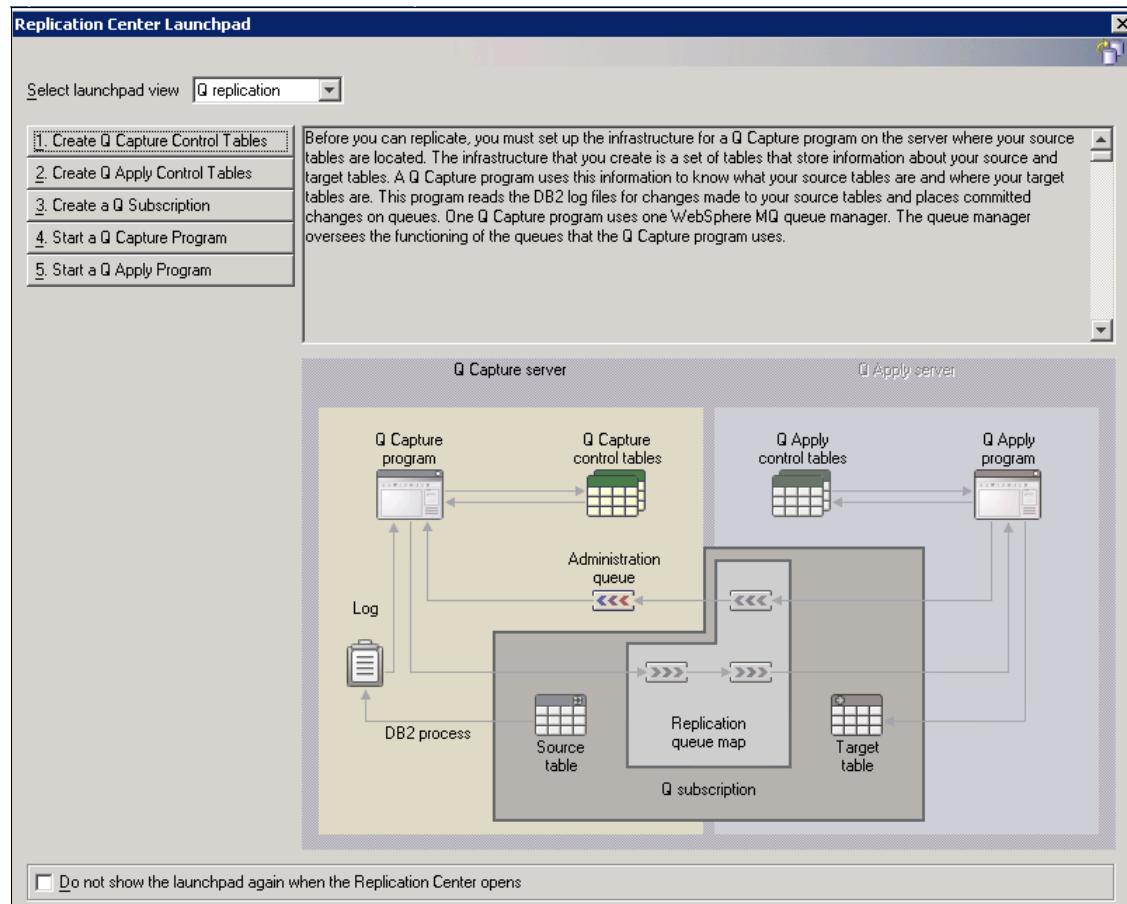


Figure 13-2 Replication Center Launchpad

- b. Click **1. Create Q Capture Control Tables** from the Replication Center Launchpad. The Create Q Capture Control Tables wizard window opens. From the first window in the wizard, which is shown in Figure 13-3, you can choose between **Typical** or **Custom** for control table creation. In our example, we select **Typical** and then click **Next**.

Creating Q Capture Control tables: When you create the Q Capture Control tables, clicking the **Custom** option allows you to define the table space or table spaces that you want the tables to be in.

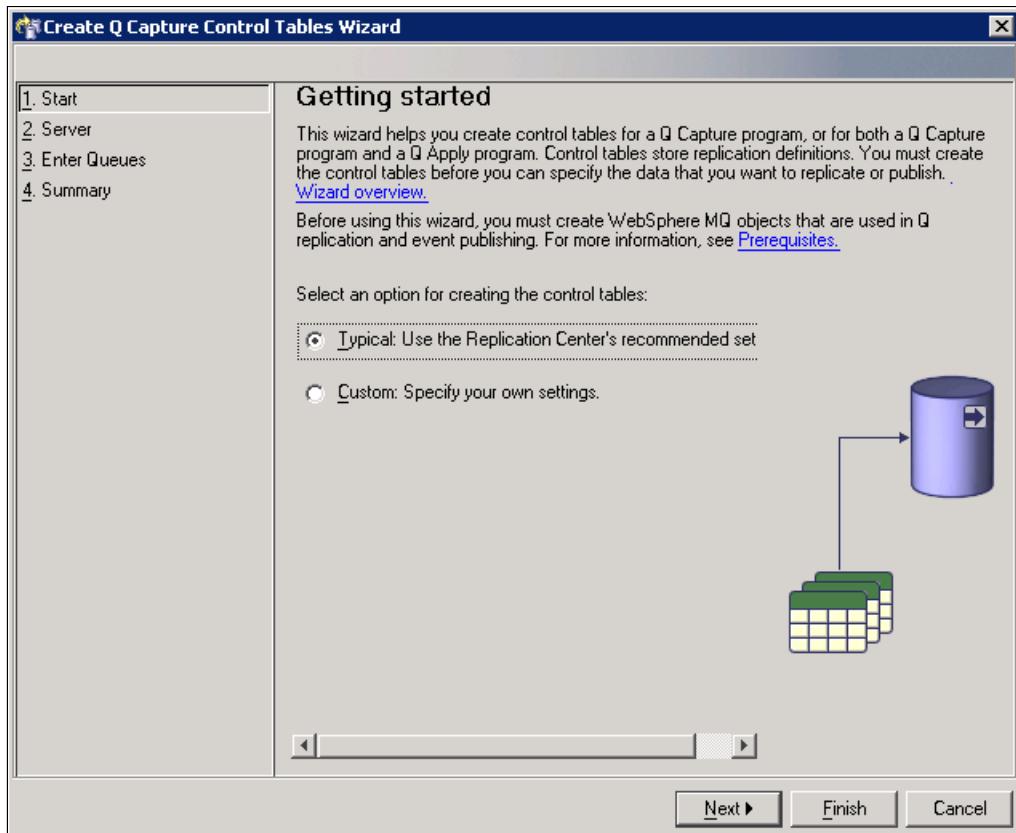


Figure 13-3 Create Q Capture Control Tables wizard - Getting started

- c. In the next window (Figure 13-4), select the Q Capture server from the list (source server). This action should populate the User ID, Password, and Q Capture schema. The default schema is ASN, but you can change it according to your client naming conventions. Click **Next** to continue.

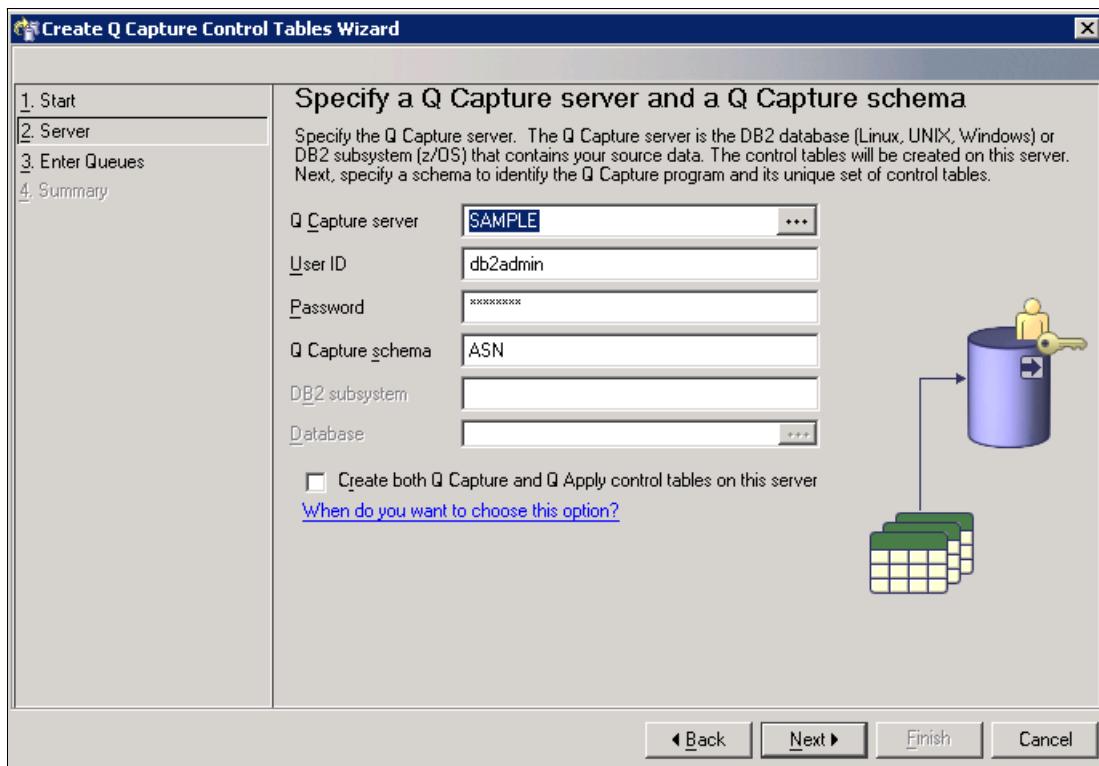


Figure 13-4 Create Q Capture Control Tables wizard - Specify a Q Capture server and Q Capture schema

- d. In the next window (Figure 13-5), enter the Queue manager name for the source server. In our example, the Queue manager is QMGR1. Enter the information for the Administration queue and Restart queue, which is SAMPLE_ADMINQ and SAMPLE_RESTARTQ in our example. Click **Next** to proceed to the summary window.

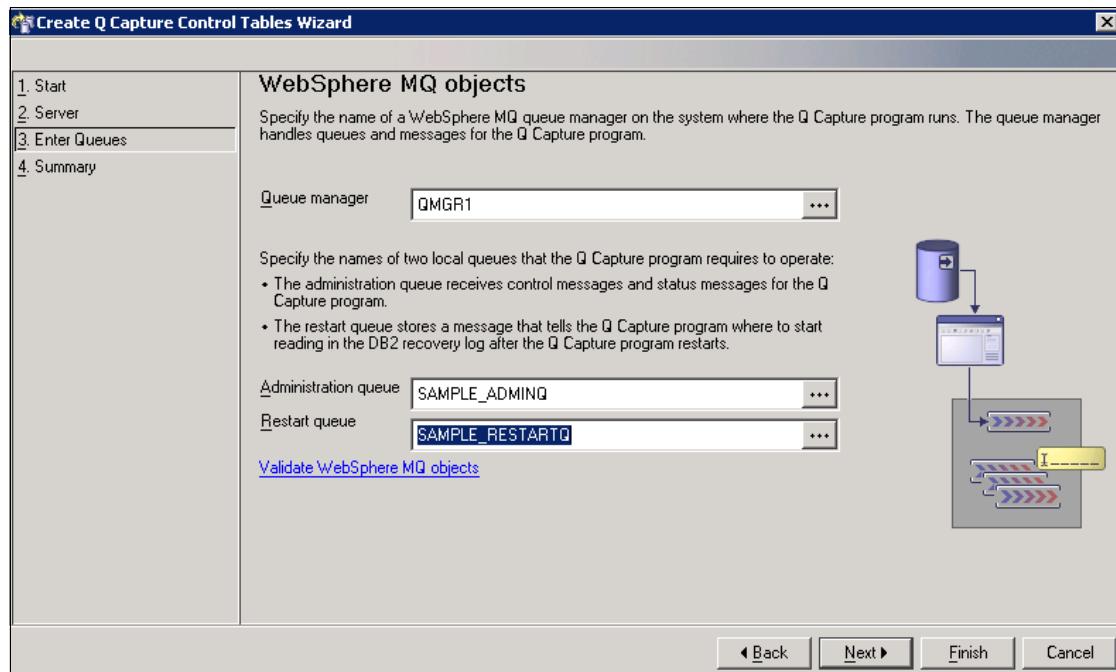


Figure 13-5 Create Q Capture Control Tables wizard - WebSphere MQ objects

- e. Review the Summary window and click **Finish** (Figure 13-6).

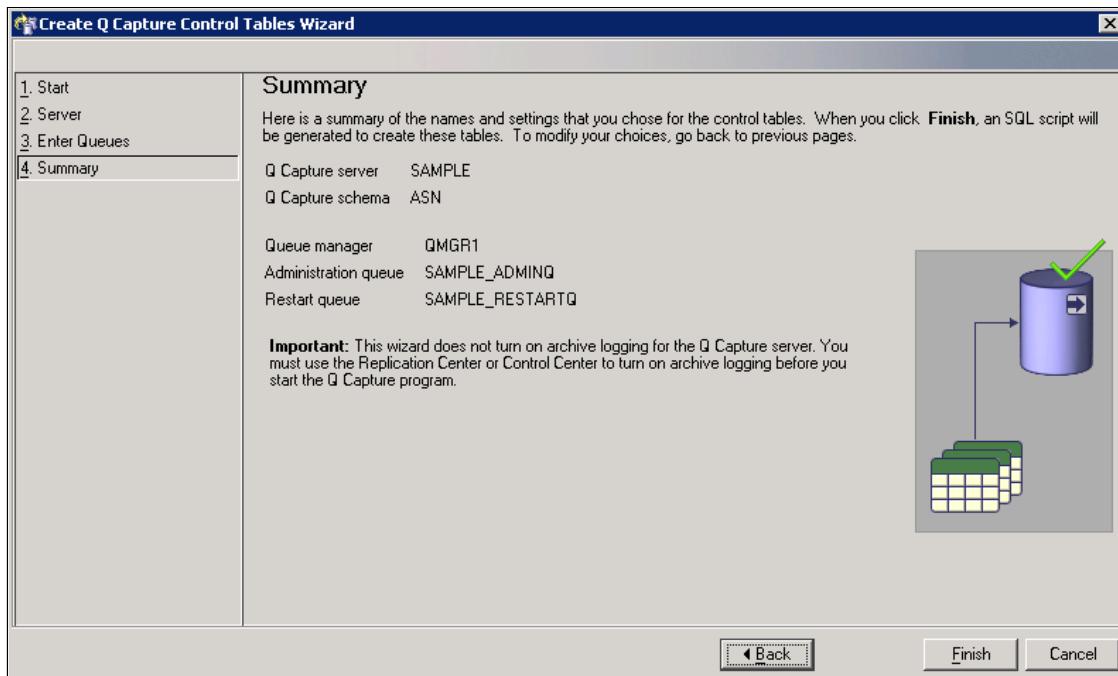


Figure 13-6 Create Q Capture Control Tables wizard - Summary

- f. In the Run Now or Save SQL window, save the script for future use, such as a rebuild. Figure 13-7 shows the script that you created and the options. After you save the SQL, select **Run now** and click **OK**.

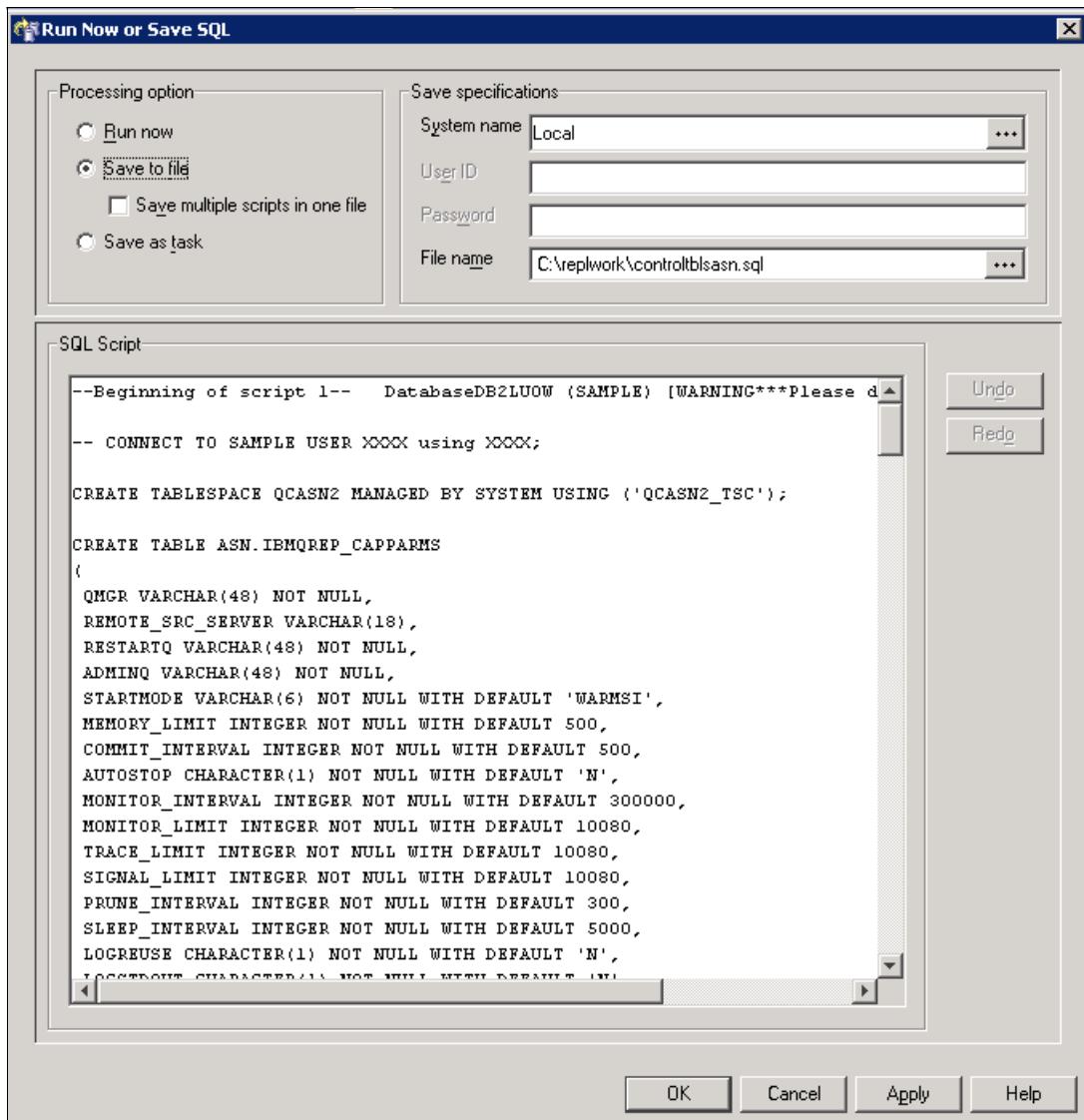


Figure 13-7 Run Now or Save SQL window for Q Capture Control Tables

6. Creating the Q Apply Control tables:

You can accomplish this task on the target or source server. If it is done from the source, ensure that the target server and database are cataloged on the source system.

Complete the following steps:

- a. Click the **Replication Center** drop-down menu and select **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 action opens the Create Q Apply Control Tables wizard. In the first window in the wizard, in our example, we select **Typical** (Figure 13-8). After you make your selection, click **Next**.

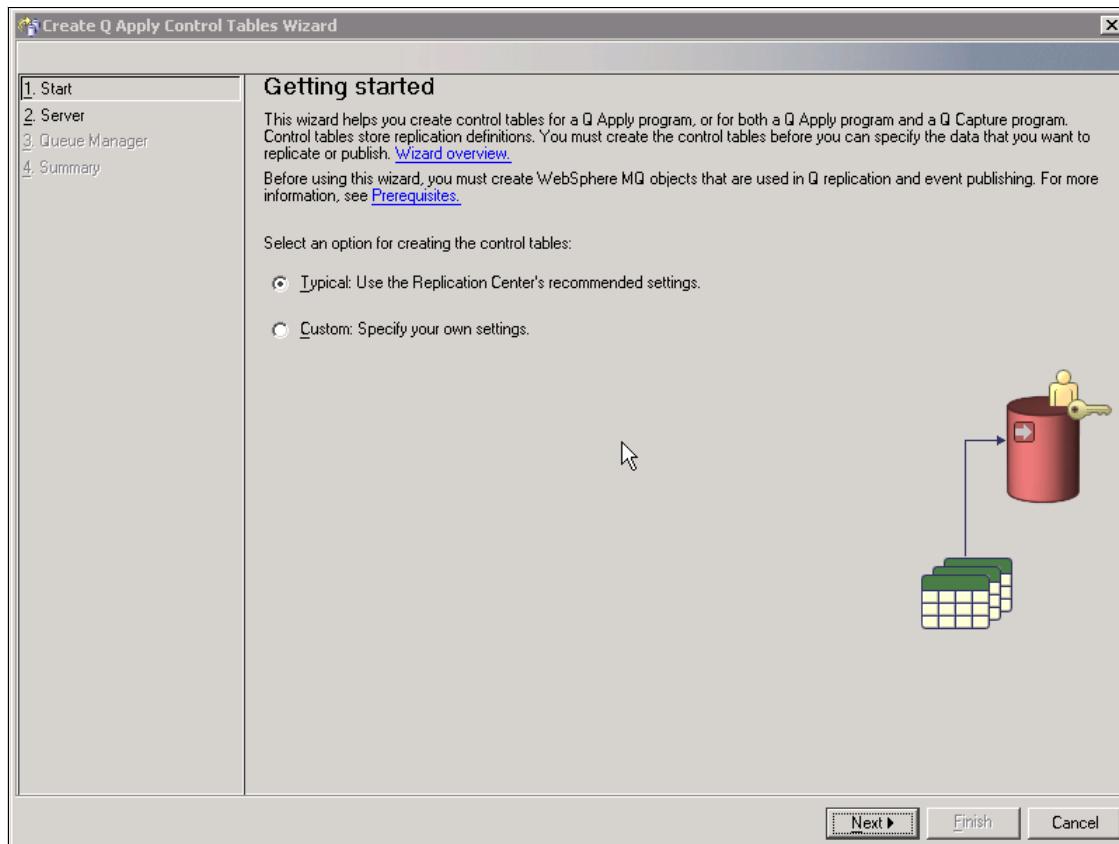


Figure 13-8 Create Q Apply Control Tables wizard - Getting started

- c. In the next window (Figure 13-9), select the Q Apply server from the list (target server). This action populates the User ID, Password, and Q Apply schema fields. The default schema is ASN, but can be changed. Click **Next** to proceed.

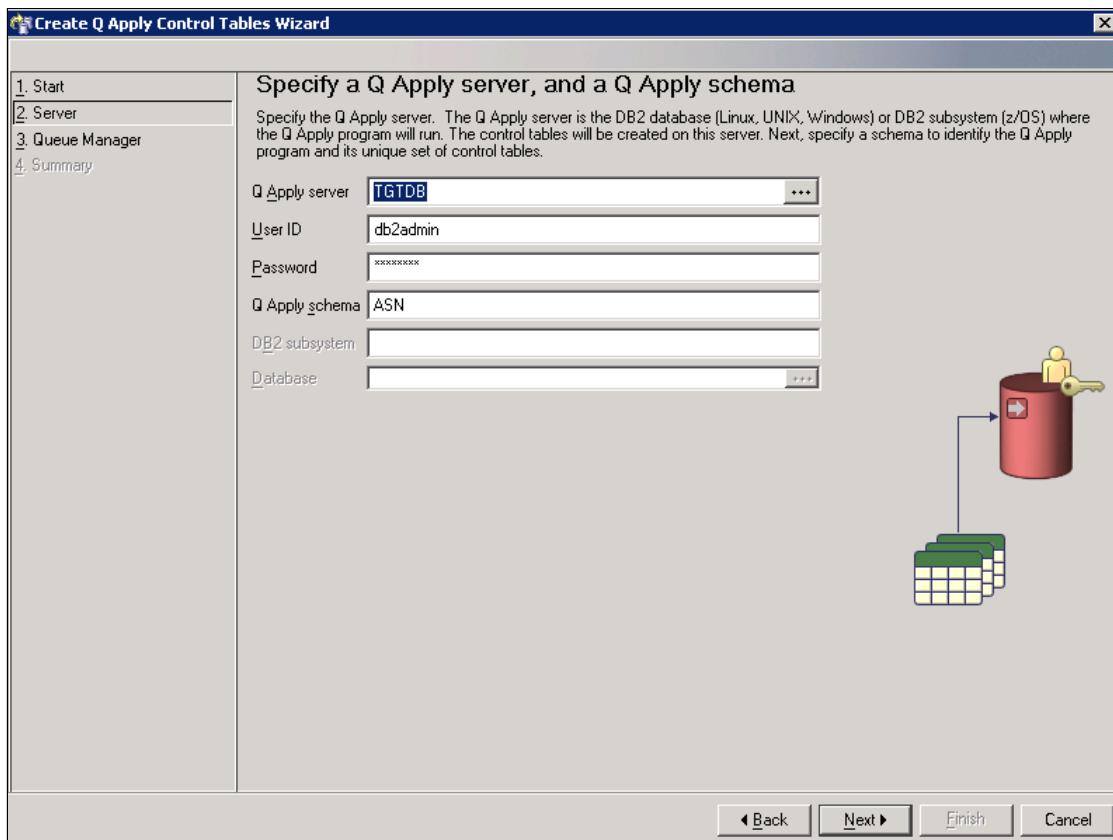


Figure 13-9 Create Q Apply Control Tables wizard - Specify a Q Apply server, and a Q Apply schema

- d. In the next window (Figure 13-10), enter the Queue manager name for the target server. In our example, the Queue manager is QMGR2. Click **Next** to proceed to the Summary window.

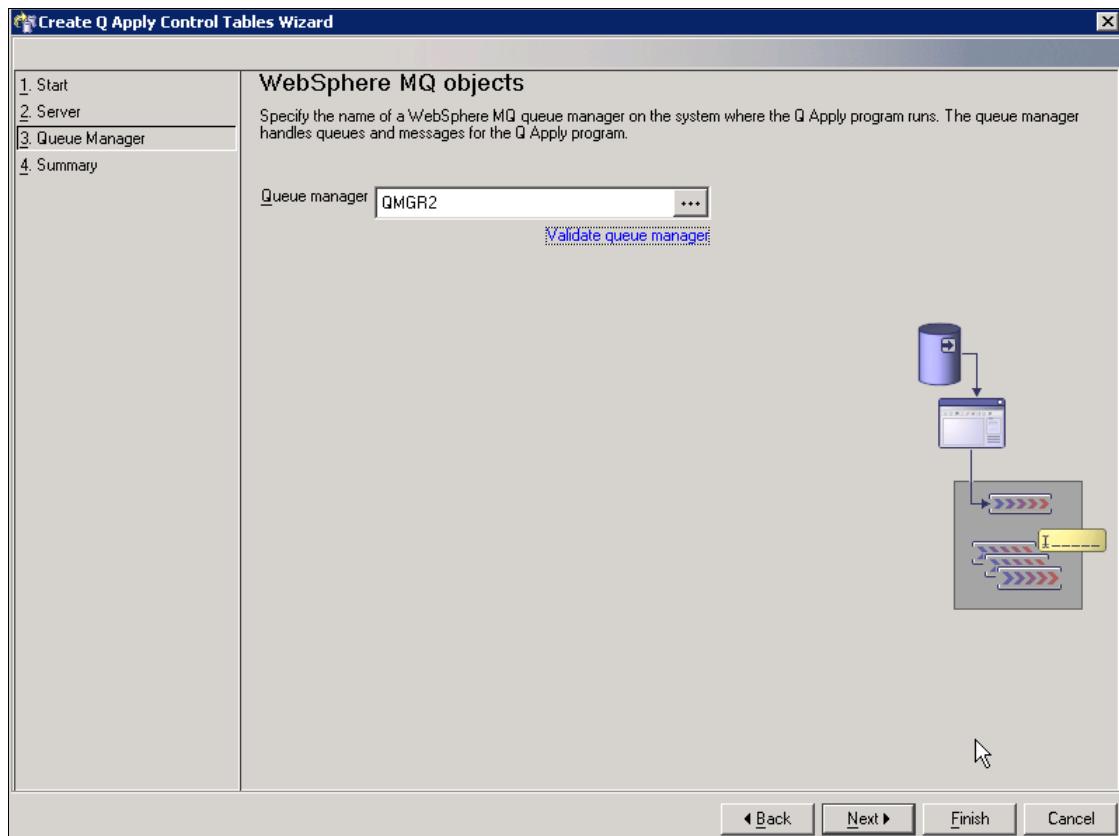


Figure 13-10 Create Q Apply Control Tables wizard - WebSphere MQ objects

- e. Review the Summary window that is shown in Figure 13-11. Click **Finish**.

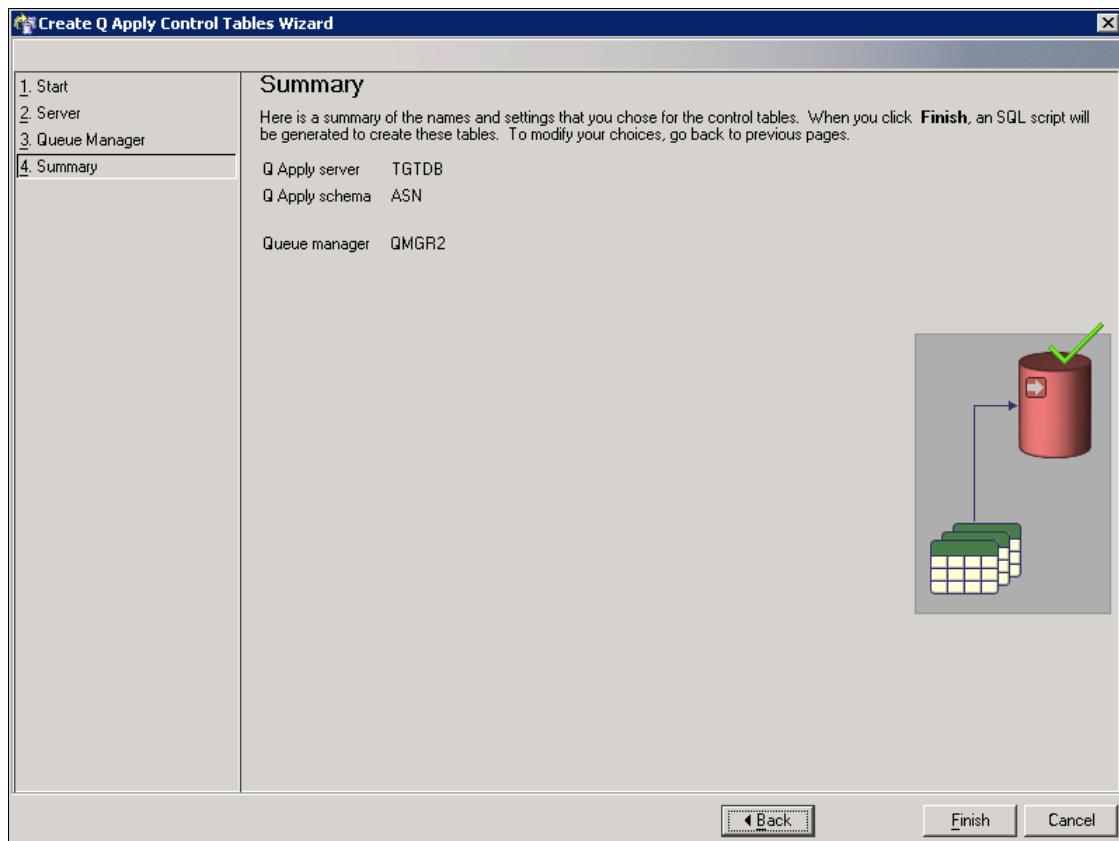


Figure 13-11 Create Q Apply Control Tables wizard - Summary

- f. In the Run Now or Save SQL window (Figure 13-12), save the script for future use, such as a rebuild. After you save the SQL, select **Run now** and click **OK**.

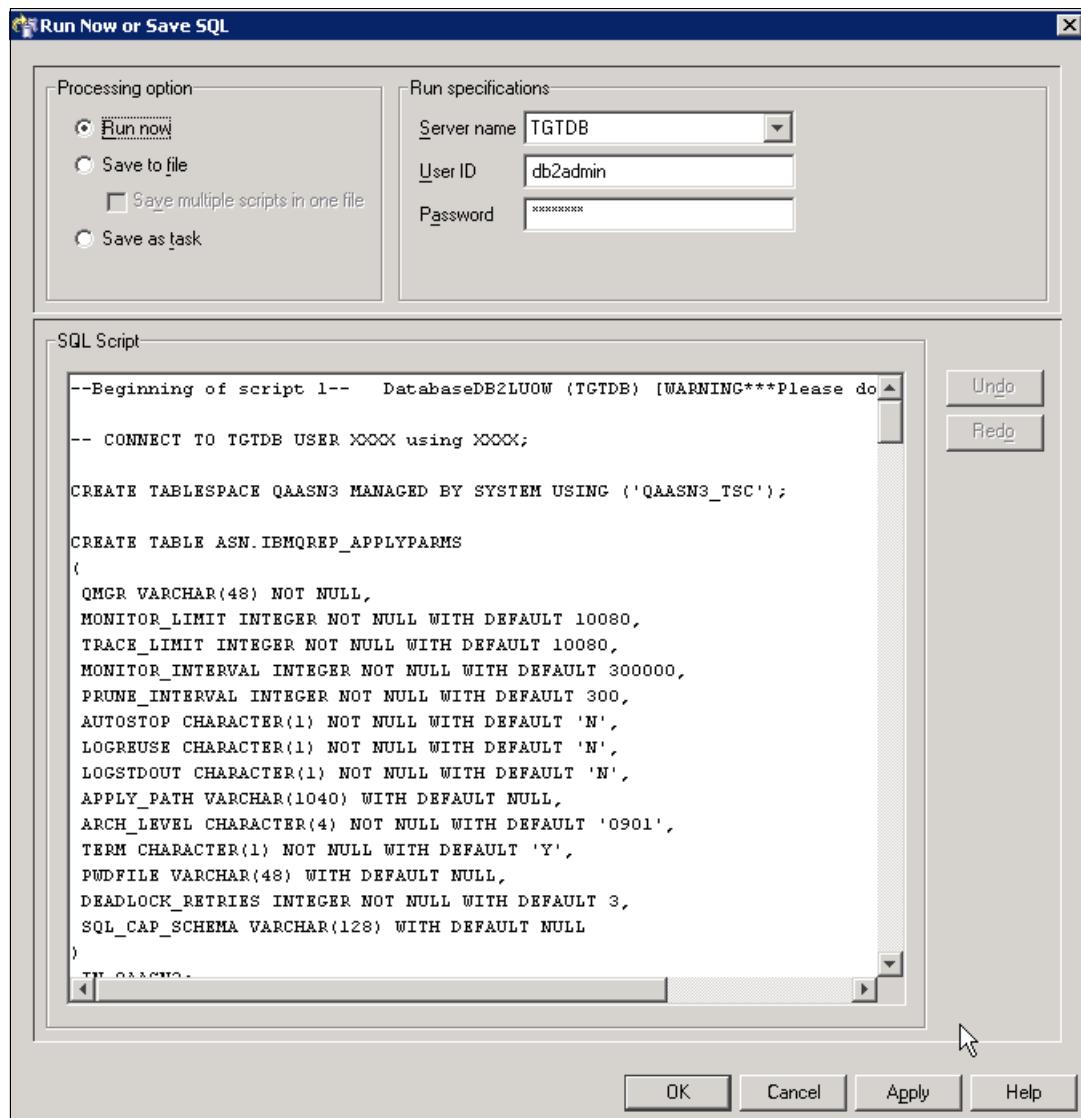


Figure 13-12 Run Now or Save SQL panel for Q Apply Control Tables

7. Create a Q subscription:

- a. From the Replication Center, click the **Replication Center** drop-down menu 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 action opens the Create a Q Subscription wizard shown in Figure 13-13. Click **Next** to continue.

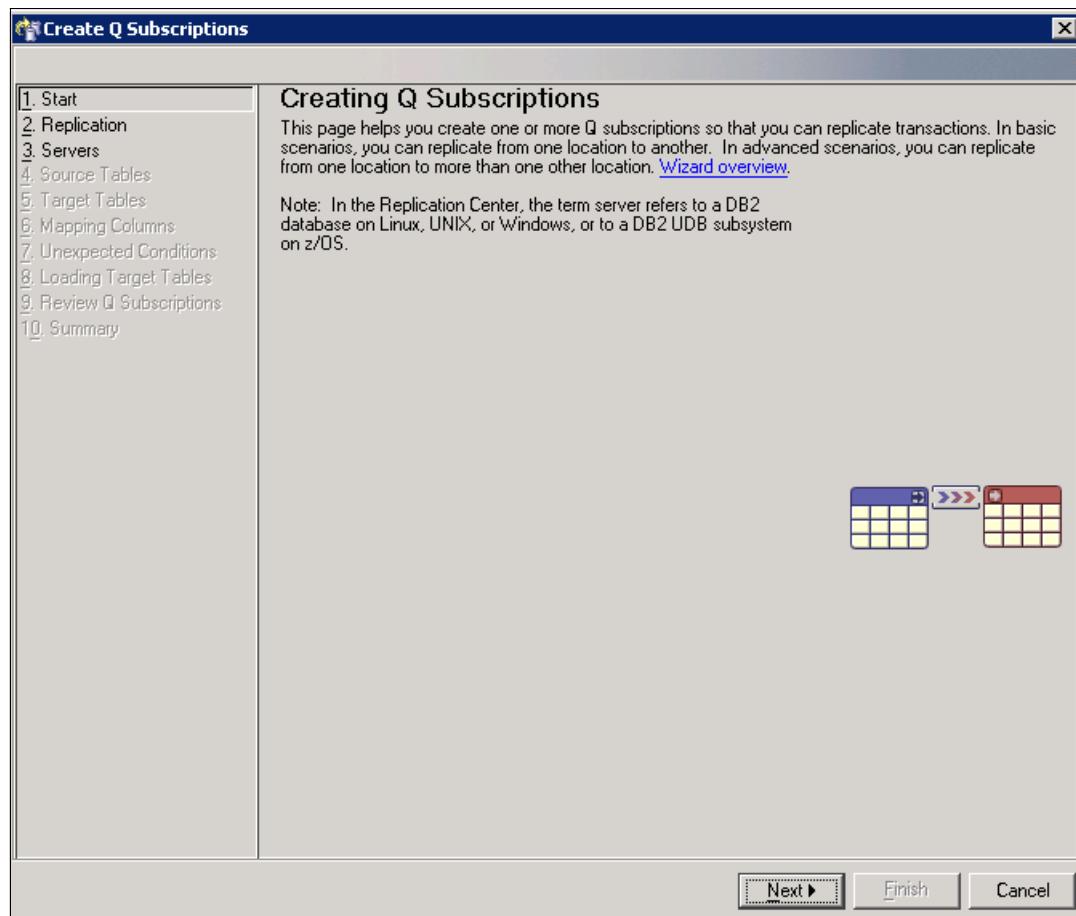


Figure 13-13 Create Q Subscriptions

- c. Choose the type of Q replication you require. In our example, we choose unidirectional replication (Figure 13-14). If you choose anything other than unidirectional, more WebSphere MQ setup steps might be required. For more information, see the DB210.1 Information Center:

<http://pic.dhe.ibm.com/infocenter/db2luw/v10r1/index.jsp?topic=/com.ibm.swg.im.iis.prod.repl.nav.doc/topics/iiyrcncccreatesubspubs.html>

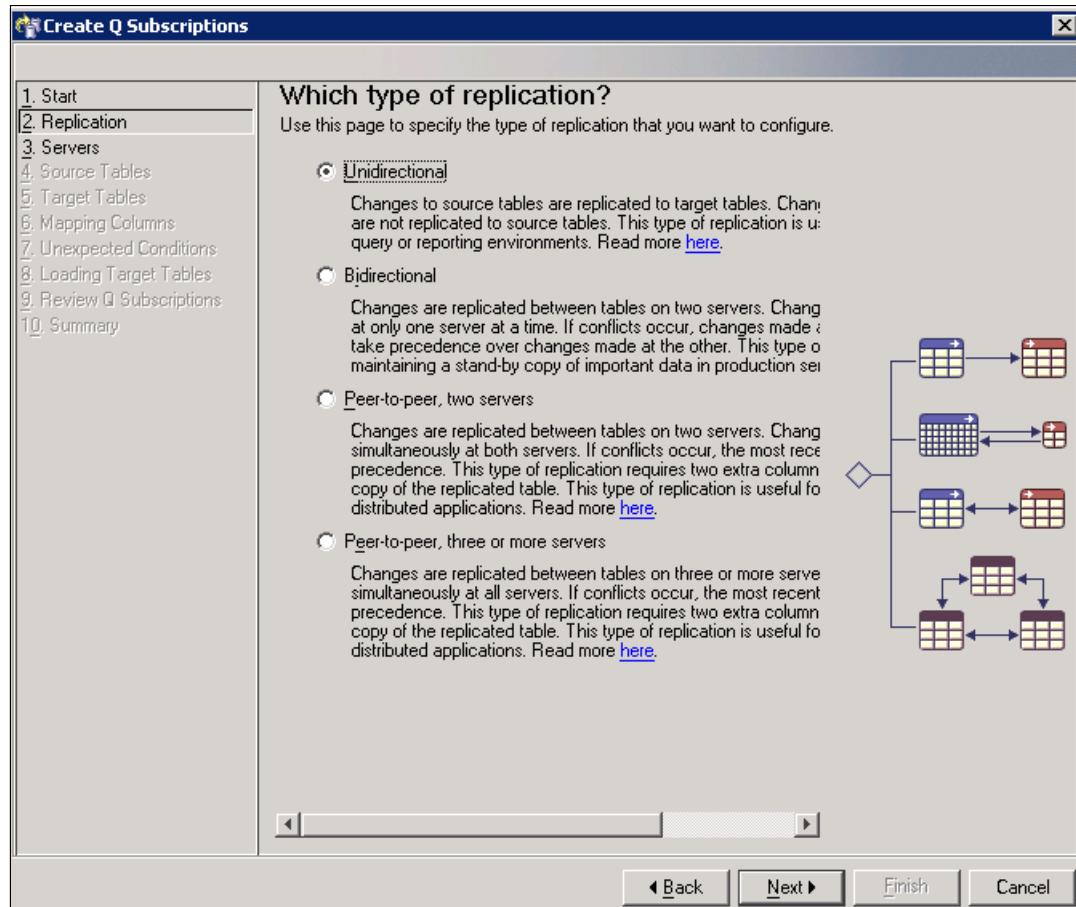


Figure 13-14 Create Q Subscriptions - Which type of replication

- d. Choose the Source and Target servers and schema (if they are not already populated). In the Queues section, click the button next to the Replication Q map field to open the Select Replication Queue Map window (Figure 13-15).

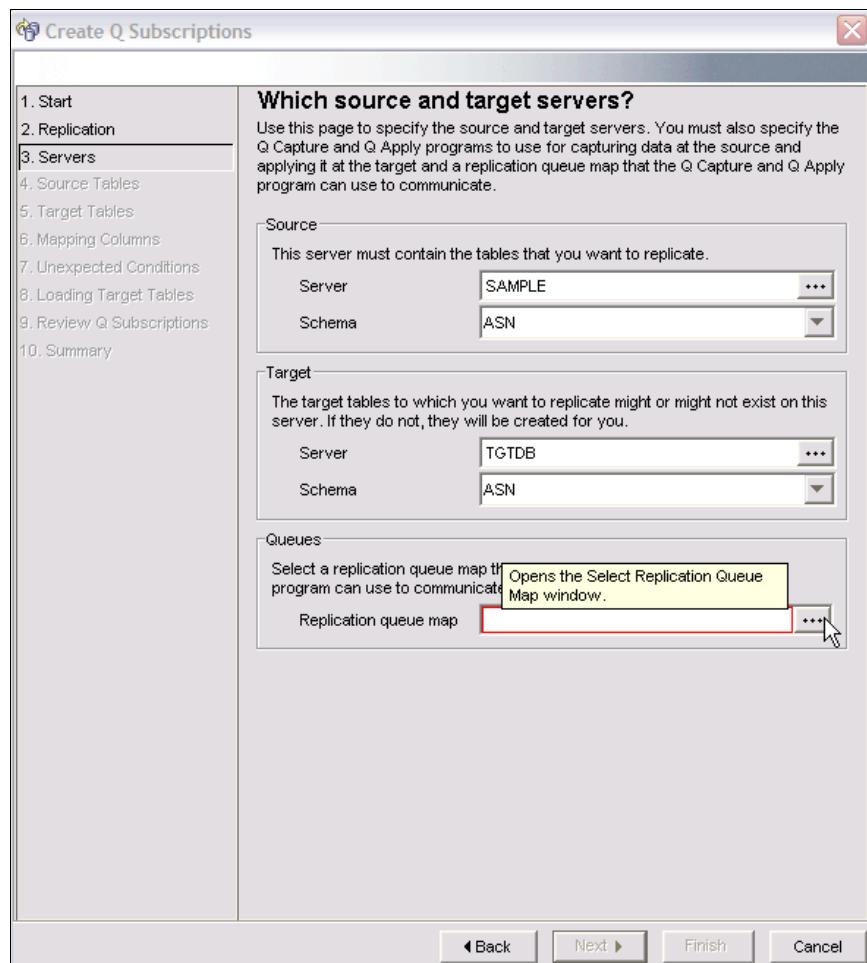


Figure 13-15 Create Q Subscriptions - Which source and target servers

- e. Click **New** to create an object (Figure 13-16).

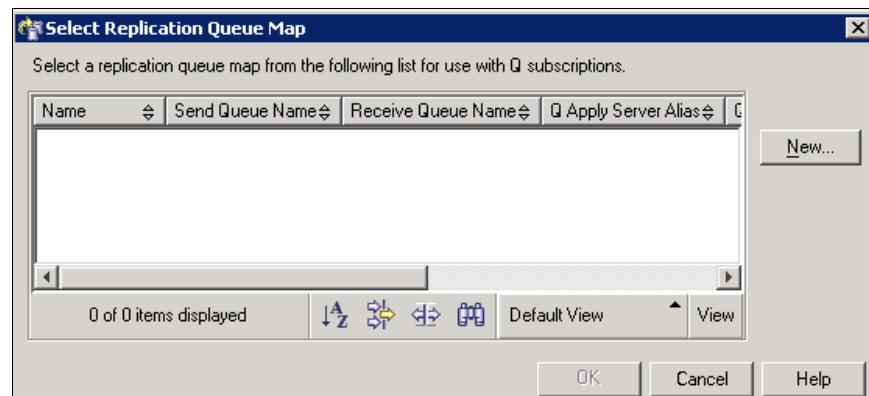


Figure 13-16 Select Replication Queue Map

- f. In the Create Replication Queue Map window, under the General tab, insert the created Send queue, Receive queue, and Administration queue. Figure 13-17 shows the object names that are entered.

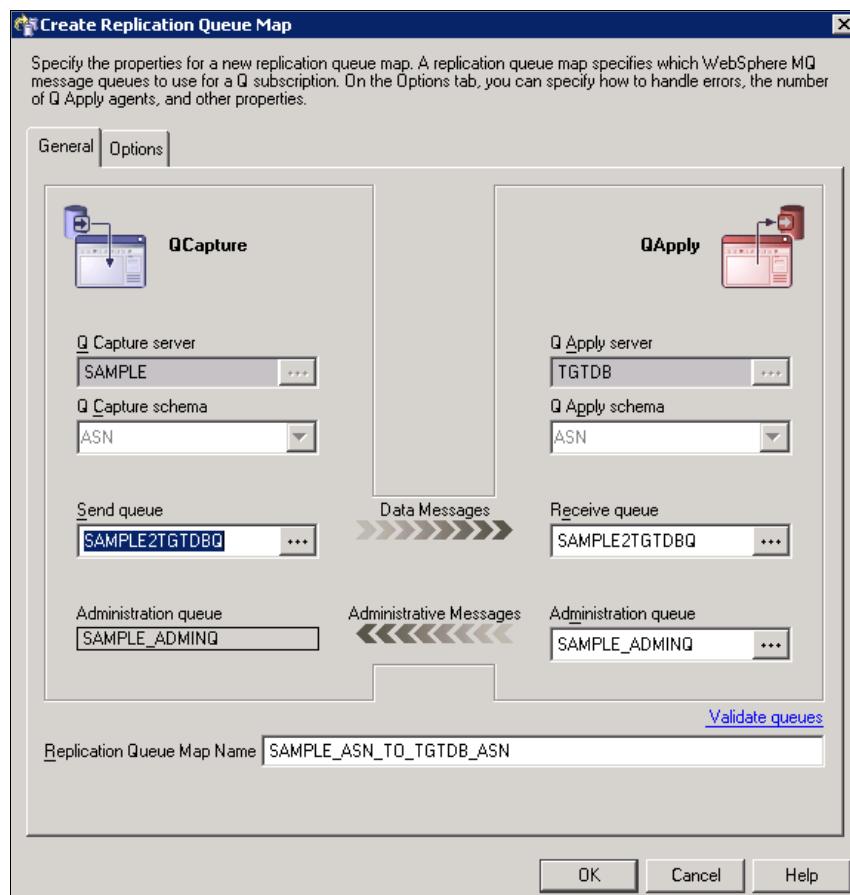


Figure 13-17 Create Replication Queue Map - General tab

- 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 (Figure 13-18). Click **OK** to finish the new replication queue map.

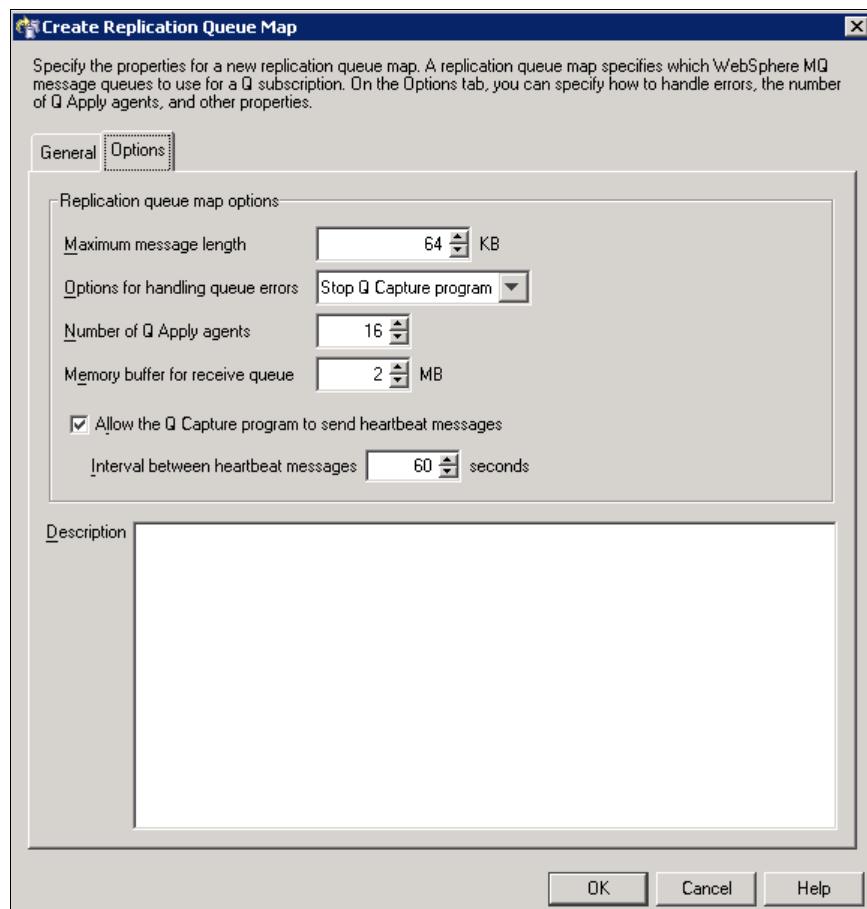


Figure 13-18 Create Replication Queue Map - Options tab

- h. In the Run Now or Save SQL window (Figure 13-19), save the script for future use, such as a rebuild. After you save the SQL, select **Run now** and click **OK**.

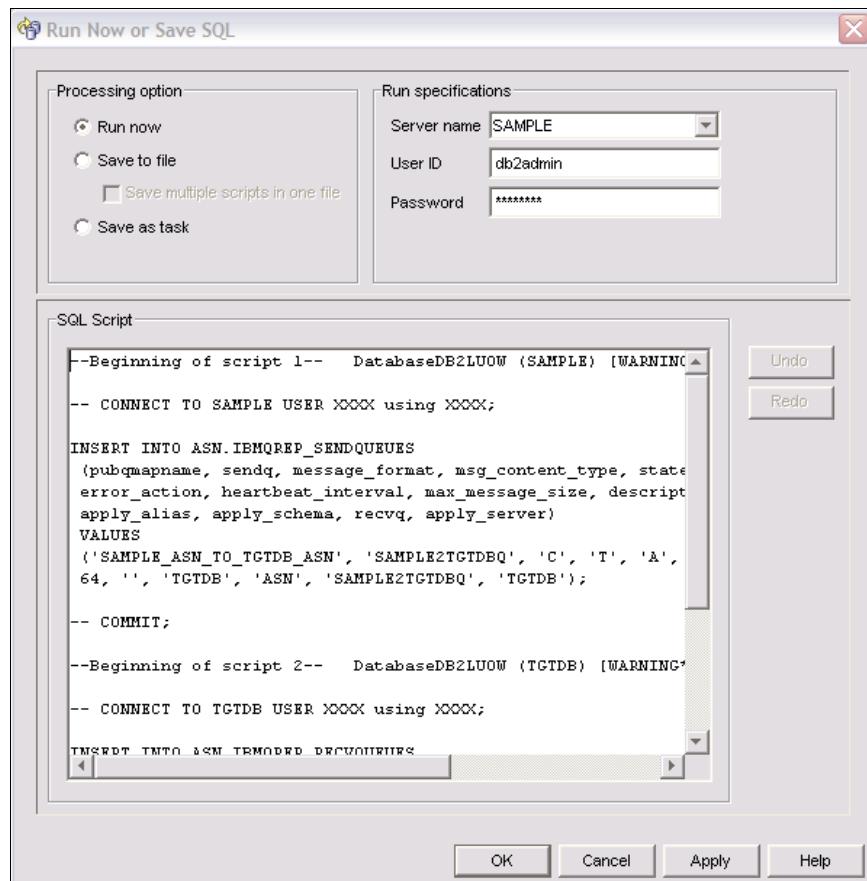


Figure 13-19 Run Now or Save SQL Queue Map

- i. From the Select Replication Queue Map, highlight your queue map and click **OK** (Figure 13-20).

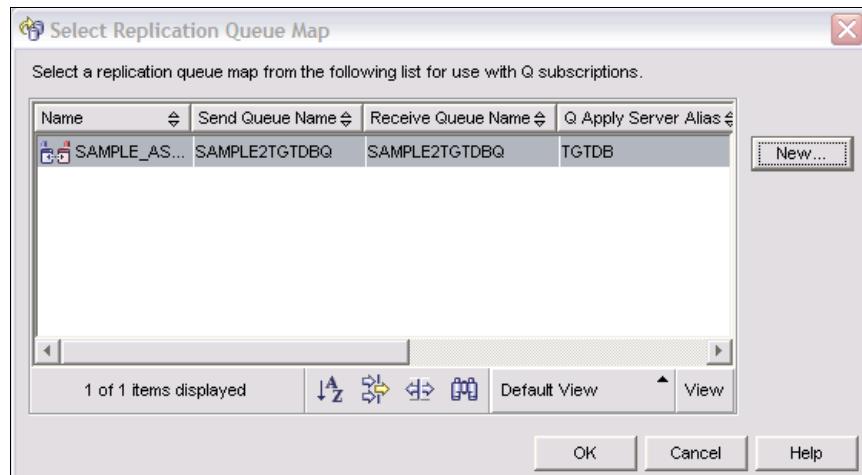


Figure 13-20 Select Replication Queue Map selected

- j. Now you should be back at the Create Q Subscription wizard window with all of the information that is entered for **3. Services** (Figure 13-21). Click **Next** to continue.

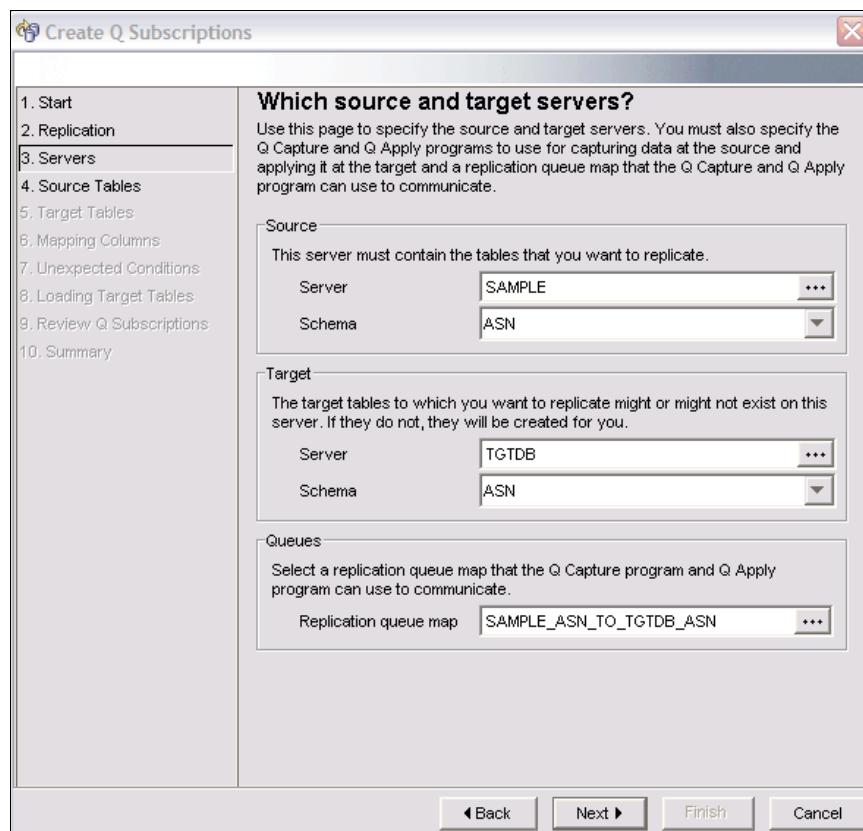


Figure 13-21 Create Q Subscriptions - Complete

- k. Select the tables from the source you want to replicate. In our example, we chose the EMPLOYEE table to replicate (Figure 13-22). Click **Next** to continue.

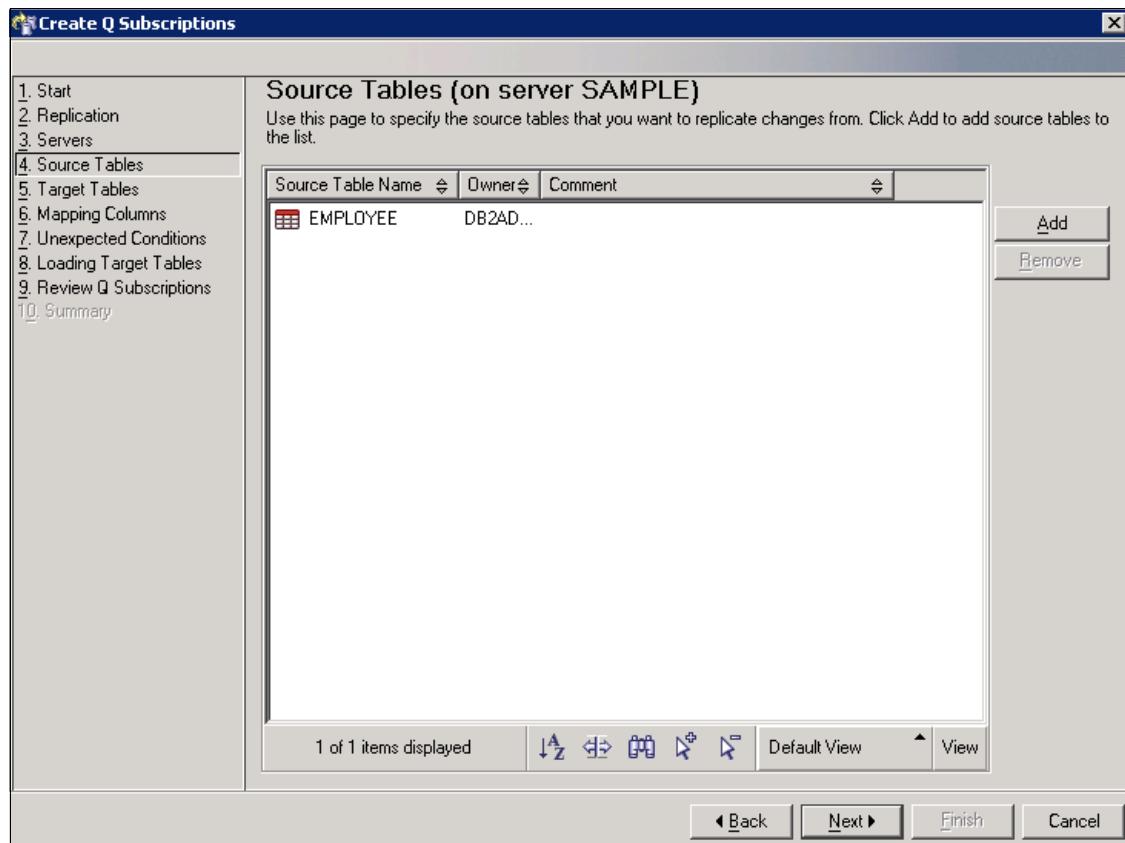


Figure 13-22 Create Q Subscriptions - Source Tables

- I. In the Create Q Subscriptions wizard, select the target type that you want. For our example, we chose to create a table on the target (Figure 13-23). Click **Next** to continue.

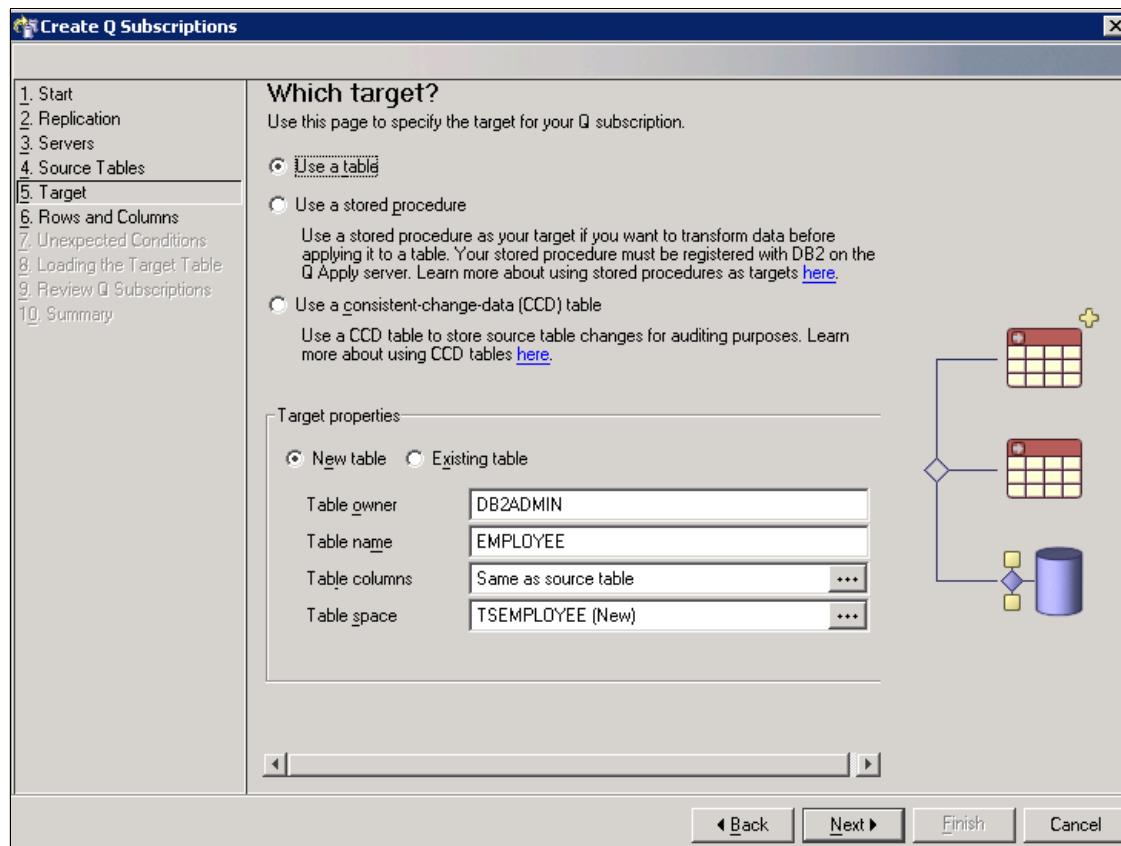


Figure 13-23 Create Q Subscriptions - target tables

- m. In the Create Q Subscriptions wizard, you can choose to use granularity with your columns and rows replicated (Figure 13-24). Click **Next** to continue.

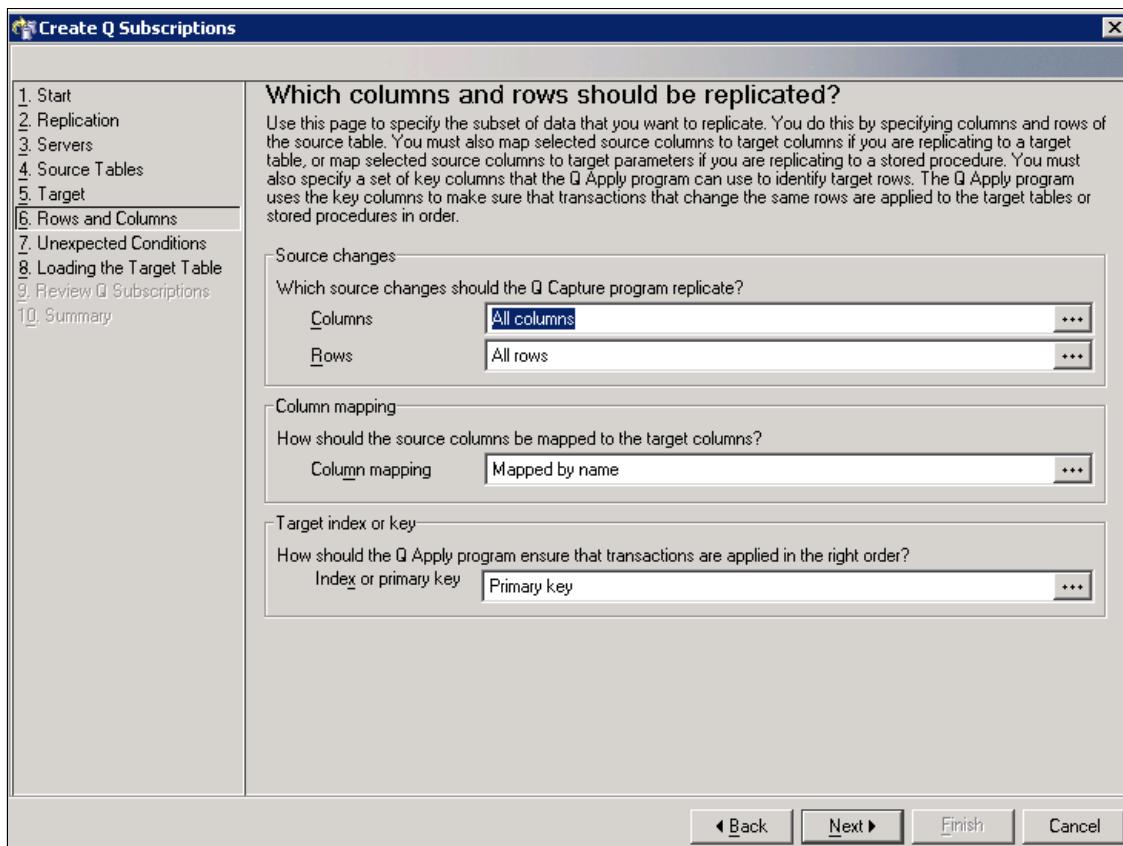


Figure 13-24 Create Q Subscriptions - replicated columns and rows

- n. In the Create Q Subscriptions wizard, you can specify how you want the Q Apply program to handle unexpected conditions (Figure 13-25). Click **Next** to continue.

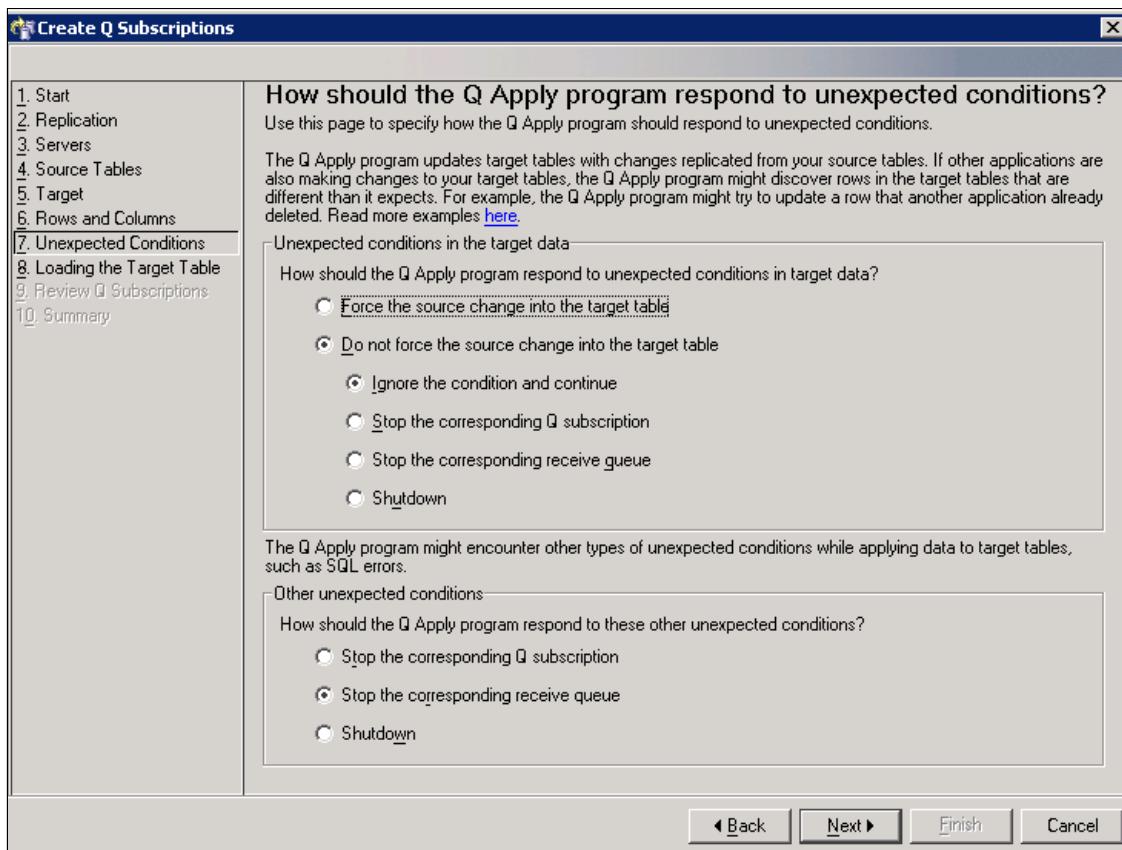


Figure 13-25 Create Q Subscriptions - unexpected conditions

- o. Specify how you want your target tables loaded (Figure 13-26). Click **Next** to continue.

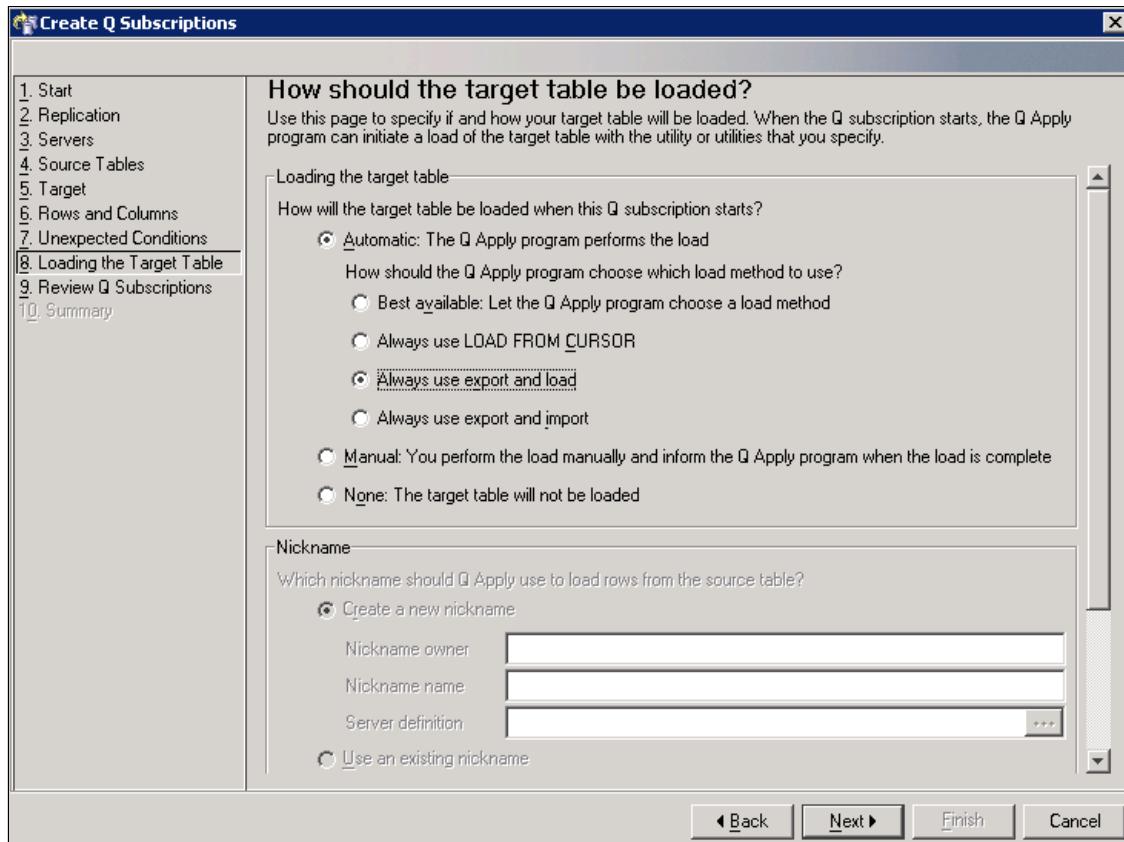


Figure 13-26 Create Q Subscriptions - loading the target table

- p. In the Create Q Subscriptions wizard, observe the settings (Figure 13-27). If they are correct, click **Next** to continue. If they are not correct, click **Back** and make any necessary changes.

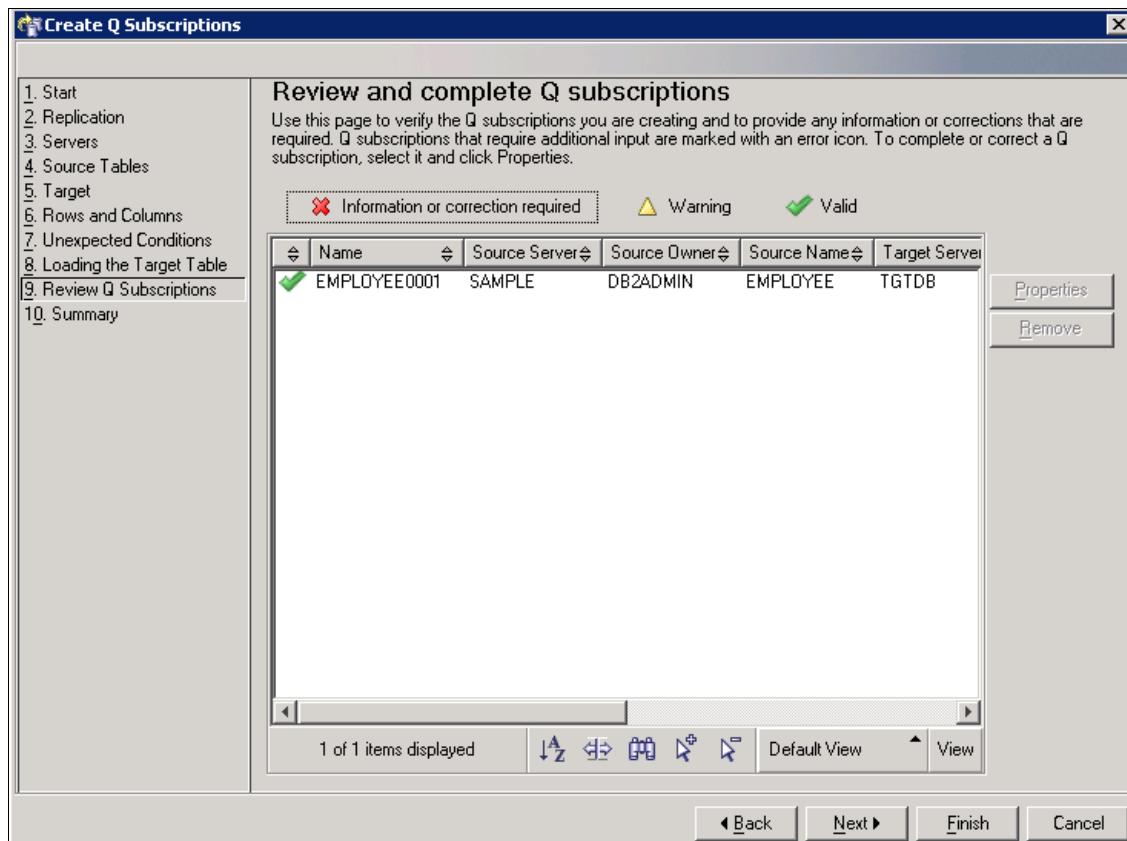


Figure 13-27 Create Q Subscriptions - review Q subscriptions

- q. In the Create Q Subscriptions wizard, the Summary window is the final window before you create the SQL script (Figure 13-28). Click **Finish** to create the SQL script.

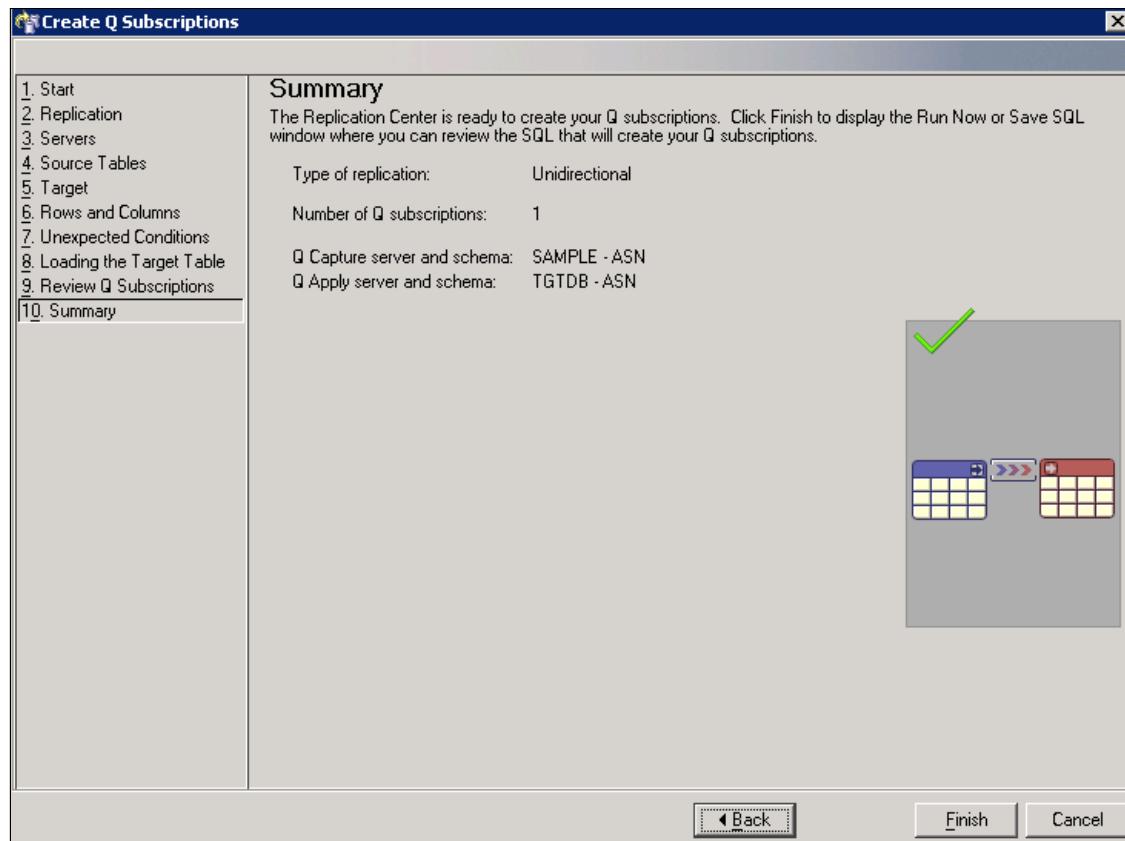


Figure 13-28 Create Q Subscriptions - Summary

- r. In the Run Now or Save SQL window (Figure 13-29), save the script for future use, such as a rebuild. After you save the SQL, select **Run now** and click **OK**.

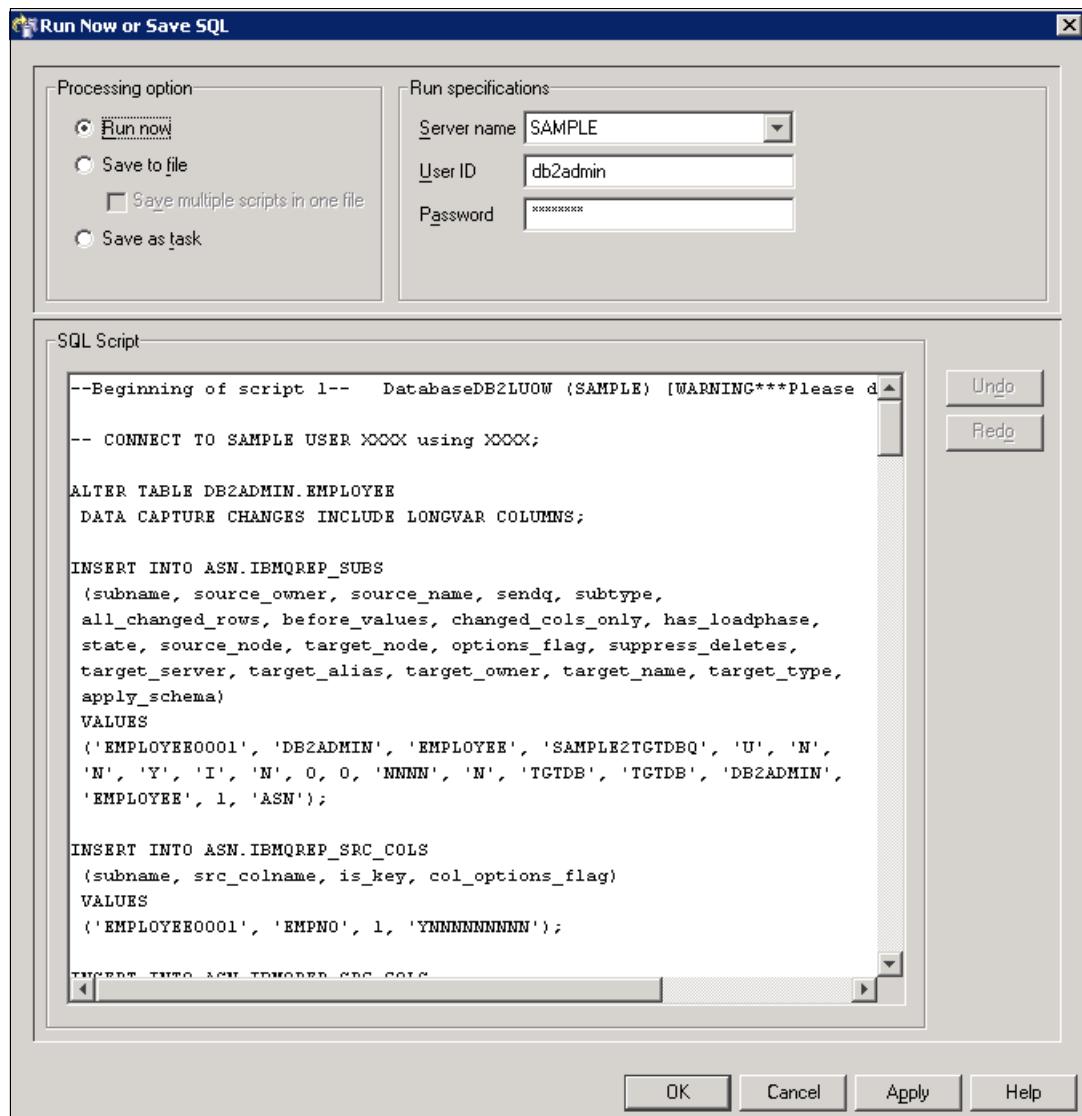


Figure 13-29 Run Now or Save SQL Q Subscriptions

8. Configure the 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.

Complete the following steps:

- a. Create a directory on the Q Apply Control server where you plan to run the Q Apply **start** command. In our example, we use `c:\replwork\`.
- b. Run the command to create the password file in the directory that you create (Example 13-5):

```
asnpwd init
```

Example 13-5 asnpwd init

```
C:\replwork> asnpwd init
2012-07-12-16.31.47.671000 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 (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
2012-07-12-16.37.10.535000 ASN1981I "Asnpwd" : "" : "Initial".
The program completed successfully using password file
"asnpwd.aut".
```

The unidirectional configuration is complete. We can start the Q replication process. When you start the Q replication, start Q Capture first, then Q Apply. To stop the replication process, stop Q Apply, then Q Capture.

13.2.1 Starting Q capture

This action 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 wanted directory and run **start** (Example 13-7):

```
asnqcap capture_server=SAMPLE
```

Example 13-7 asnqcap

```
C:\replwork> asnqcap capture_server=SAMPLE
2012-07-12-16.49.51.309000 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.
2012-07-12-16.49.51.329000 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 (Figure 13-30).

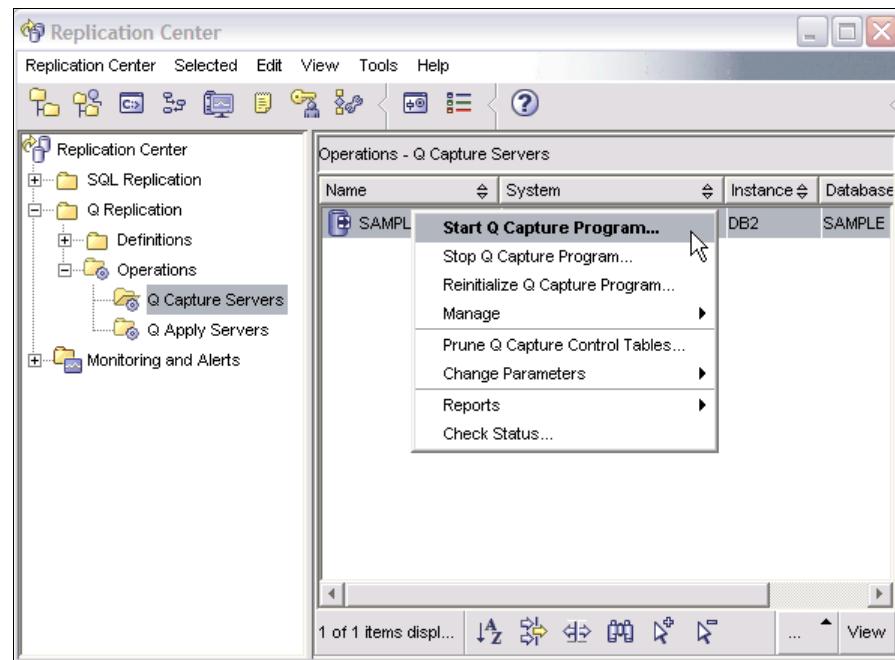


Figure 13-30 Replication Center - start Q Capture

2. From the Run Now or Save Command window, enter the required information in to the User ID, Password, and start Directory fields. Click **OK** to run the command (Figure 13-31).

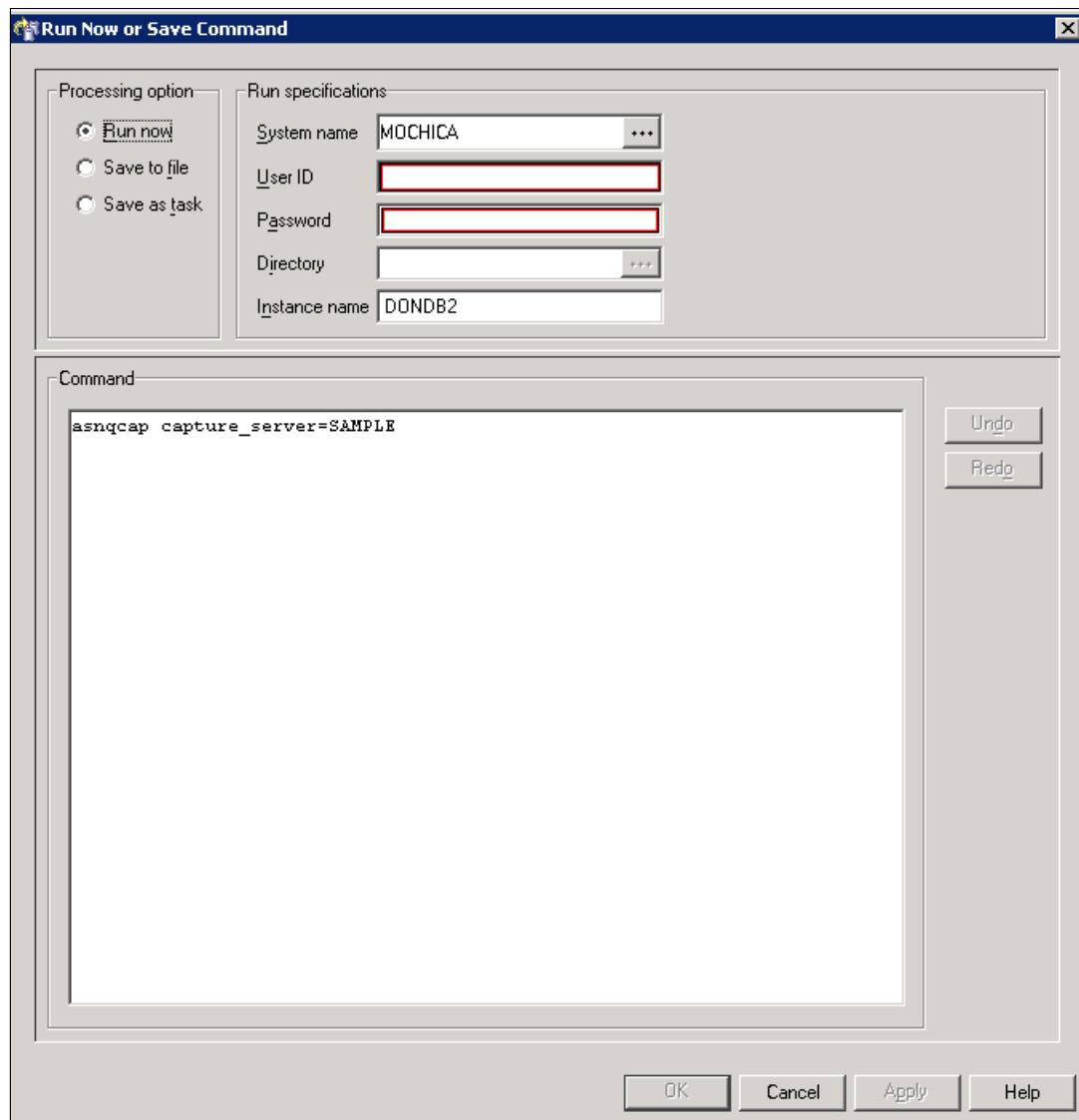


Figure 13-31 Run Now or Save Command - Q Capture Start

13.2.2 Start Q Apply

This action 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 wanted directory and run **start** (Example 13-8):

```
asnqapp apply_server=TGTDB
```

Example 13-8 asnqapp

```
C:\replwork> asnqapp apply_server=TGTDB
2012-07-12-15.58.20.785000 ASN7526I "Q Apply" : "ASN" : "BR00000" :
The Q Apply program has started processing the receive queue
"SAMPLE2TGTDBQ" for replication queue map "SAMPLE ASN TO TGTDB ASN".
```

Replication Center

To start Q Apply, complete 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 (Figure 13-32).

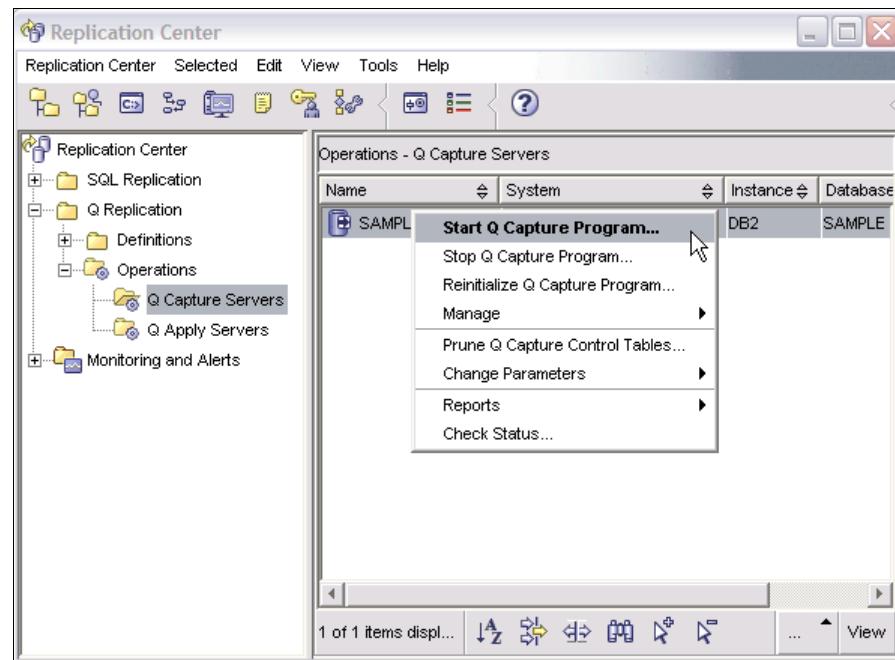


Figure 13-32 Replication Center - start Q Apply

2. From the Run Now or Save Command window, enter the required information in to the User ID, Password, and start Directory fields. Click **OK** to run the command. (Figure 13-33).

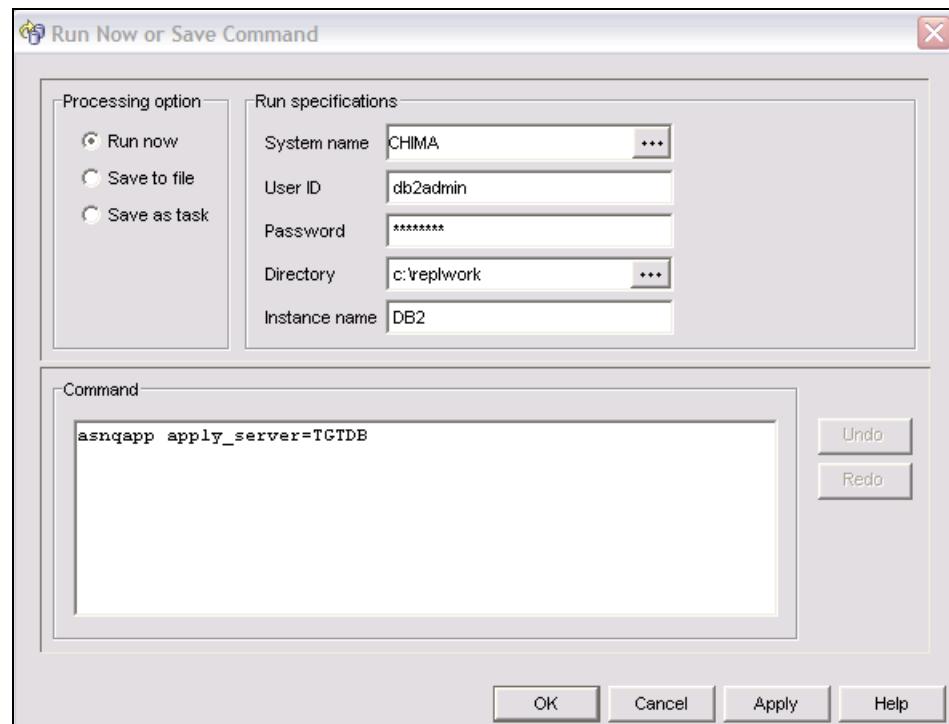


Figure 13-33 Run Now or Save Command - start Q Apply



IBM InfoSphere Change Data Capture

In this chapter, we provide an introduction to IBM InfoSphere Change Data Capture (InfoSphere CDC) and an overview of its components and modes of operation.

This chapter covers the following topics:

- ▶ Introduction
- ▶ Architectural overview
- ▶ InfoSphere CDC topologies
- ▶ Features and functionality

14.1 Introduction

InfoSphere CDC is a replication solution that captures and delivers database changes dynamically. InfoSphere CDC allows replication between heterogeneous environments, supporting various operating systems and databases. Destinations for InfoSphere CDC replication include databases (which can be different from the source database), message queues, or an extract, transform, and load (ETL) solution, such as IBM InfoSphere DataStage.

What is replicated depends on the table mappings that are configured in the InfoSphere CDC Management Console GUI application. InfoSphere CDC employs a non-invasive approach to capturing changes that take place on the source database, reading changes directly from database logs. No changes are required to the source application. The capture mechanism (*log scraping*) is a lightweight process that runs on the source server and avoids significant impact on production systems.

InfoSphere CDC also helps reduce processing impact and network traffic by sending only the data that change. Changes are sent from source to target through a standard Internet Protocol network socket. Replication can be carried out continuously or periodically. When data is transferred from a source server, it can be remapped or transformed in the target environment. With a database as the target for InfoSphere CDC, changes are applied through standard SQL statements.

In the rest of this chapter, we provide an architectural overview of InfoSphere CDC, insight into its different implementation topologies, and a closer look at its features and functionalities.

For more information about InfoSphere CDC, see the InfoSphere CDC Information Center at:

<http://publib.boulder.ibm.com/infocenter/cdc/v6r5m1/index.jsp>

14.2 Architectural overview

This section presents the building blocks of InfoSphere CDC, and highlights the mechanisms that are employed to ensure transaction integrity and reliable performance.

14.2.1 InfoSphere CDC architecture

Figure 14-1 provides an overview of the InfoSphere CDC architecture. InfoSphere CDC has distinct software editions that are optimized for various hardware platforms, operating system environments, and database vendors. An InfoSphere CDC instance can operate as a replication source, replication target, or simultaneously as a replication source and target. An InfoSphere CDC instance can connect only to a single database, but more than one InfoSphere CDC instance can connect to the same database. Sources and targets for InfoSphere CDC replication are referred to as *data stores*. Data stores are logical entities that represent the data files and processes required to accomplish data replication. Each data store represents the database or target destination to which you want to connect.

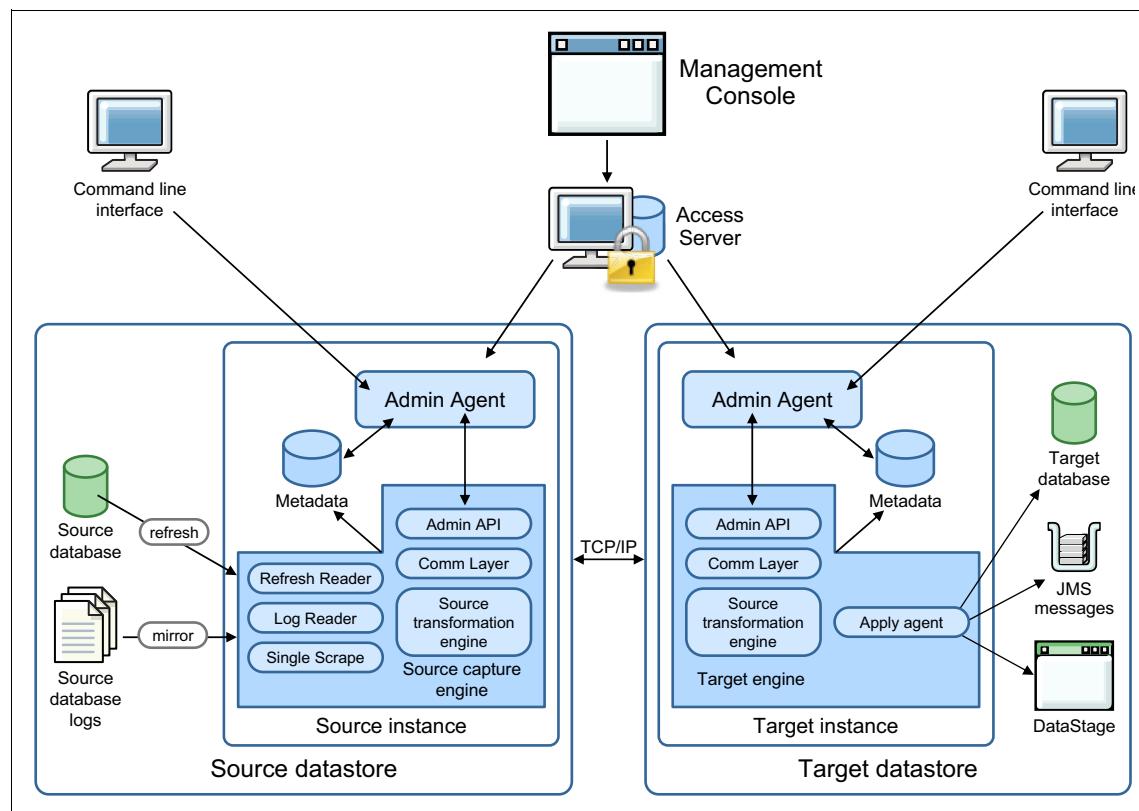


Figure 14-1 InfoSphere CDC architectural overview

The key components of the InfoSphere CDC architecture are:

- ▶ Management Console

An administration GUI application that you can use to configure and monitor replication. You can use Management Console to manage replication on various servers, specify replication parameters, and initiate refresh and mirroring operations. Management Console can run only on Microsoft Windows platforms, and can either be installed on the same server with other components of the InfoSphere CDC solution (Windows platforms only) or separately on a client workstation. After you define the data that be replicated and starting replication, you can close Management Console on the client workstation without affecting data replication activities between source and target servers. Without the Management Console, InfoSphere CDC can be administered through the command-line interface.
- ▶ Access Server

A server application that directs communications between Management Console and replication engine processes, controlling all non-command line access to the replication environment. When you log in to Management Console, you are connecting to Access Server. Access Server can be closed on the client workstation without affecting active data replication activities between source and target servers.
- ▶ Command-line interface (CLI)

You can use the CLI to administer data stores and user accounts, and perform administration scripting, independent of Management Console.
- ▶ Source and target data stores

Represent the data files and InfoSphere CDC instances that are required for data replication. Each data store represents a database to which you want to connect and acts as a container for your tables. Tables made available for replication are contained in a data store.
- ▶ Metadata

Represents the information about the relevant tables, mappings, subscriptions, notifications, events, and other particulars of a data replication instance that you set up.
- ▶ Replication engine

Sends and receives data. The process that sends replicated data is the *Source Capture Engine* and the process that receives replicated data is the *Target Engine*. An InfoSphere CDC instance can operate as a source capture engine and a target engine simultaneously.

- ▶ Admin API
 - Operates as an optional Java based programming interface that you can use to script operational configurations or interactions.
- ▶ Source database logs
 - Maintained by the source database for its own recovery purposes. The InfoSphere CDC log reader inspects these logs in the mirroring process, but filters out the tables that are not in scope for replication.
- ▶ Single scrape
 - Acts as a source-only log reader and a log parser component. It checks and analyzes the source database logs for all of the subscriptions on the selected data store.
- ▶ Source transformation engine
 - Processes row filtering, critical columns, column filtering, encoding conversions, and other data to propagate to the target data store engine.
- ▶ Target transformation engine
 - Processes data and value translations, encoding conversions, user exits, conflict detections, and other data on the target data store engine.
- ▶ Apply agent
 - Acts as the agent on the target that processes changes as sent by the source.

There are two replication methods that are supported with InfoSphere CDC:

- ▶ Refresh
 - Performs the initial synchronization of the tables from the source database to the target. This method is read by the Refresh reader.
- ▶ Mirror
 - Performs the replication of changes to the target table or accumulation of source table changes used to replicate changes to the target table later. InfoSphere CDC supports bidirectional replication, which allows mirroring to occur simultaneously to and from both the source and target tables.

There are three types of target-only destinations for replication that are not databases:

- ▶ JMS Messages
 - Acts as a JMS message destination (queue or topic) for row-level operations that are created as XML documents.

- ▶ Flat files
Act as an intermediary stage for subsequent ETL processing.
- ▶ InfoSphere DataStage
Processes changes delivered from InfoSphere CDC that can be used by InfoSphere DataStage jobs.

14.2.2 Transactional integrity and reliability

Transactional consistency is maintained by applying changes on the target database in the same order they are run on the source database. Tables with referential relationships must be incorporated within the same replication subscription to ensure that referential integrity is maintained. More information about table subscriptions is in the next section.

With relational database management systems (RDBMSes), all changes that are made to the database system are written to the database log, including uncommitted data that may be committed or rolled back. InfoSphere CDC monitors these uncommitted changes at the source database, maintaining changes that are associated with each open transaction in a separate queue. These InfoSphere CDC queues are called *Transaction Queues*. Whenever a commit is seen, associated changes are sent to the target database. If a rollback is issued instead, associated changes are removed from the Transaction Queue.

Transaction Queues are shown in Figure 14-2.

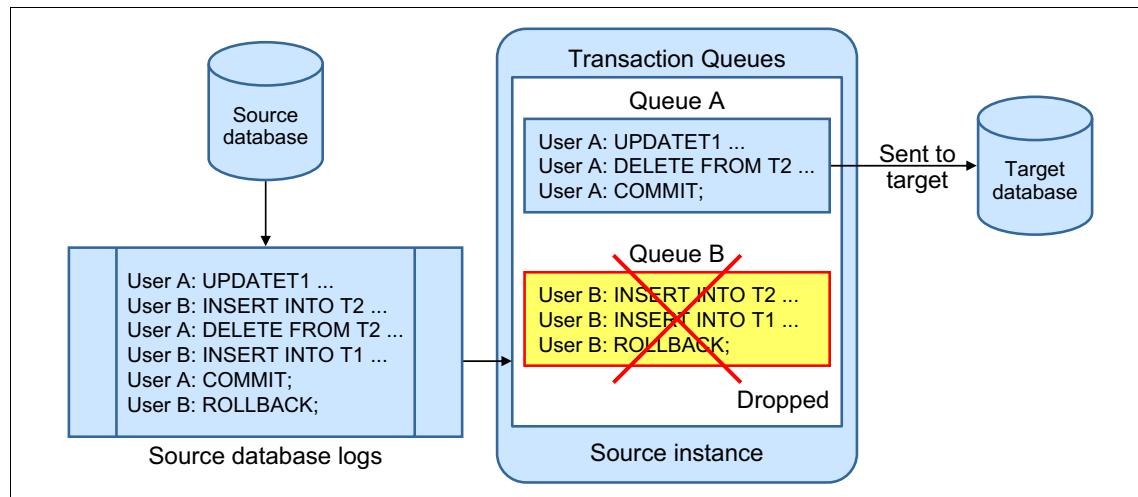


Figure 14-2 Transaction Queues

Bookmarks

InfoSphere CDC uses bookmarks to be able to recover from network or database outages. Bookmarks enable InfoSphere CDC to maintain transaction tracking by regularly storing information about successfully replicated transactions at the target database.

Bookmarks mark points in the flow of replicated transactions, and contain all the information InfoSphere CDC needs to continue replication after a normal (planned) or abnormal (outage) interruption. Successfully received and applied transactions have their bookmarks stored in a metadata table in the target database. This metadata table is updated within the same unit of work with the applied transaction.

Whenever replication is restarted, the target InfoSphere CDC instance communicates the latest bookmark information to the source instance, and so replication can continue without resending previously sent changes, and without any data loss.

14.3 InfoSphere CDC topologies

InfoSphere CDC supports various solution topologies that cater to different business needs. The flexibility of InfoSphere CDC with regards to database systems, hardware platforms, and operating system environments is of particular value for replication solutions that involve existing production systems. In this section, InfoSphere CDC data stores, subscriptions, and table mappings are introduced. Next, we cover some of the replication topologies and schemes that are supported by InfoSphere CDC.

- ▶ Data stores

Data stores are logical entities that represent the data files and processes required to accomplish data replication. Each data store represents the database or target destination to which you want to connect.

- ▶ Subscriptions

A subscription is a connection that is required to replicate data between a source data store and a target data store. It contains details about the data that is being replicated and how the source data is applied to the target. Subscriptions use data stores as the source or target of replicated data. Multiple subscriptions can be defined for each InfoSphere CDC instance. Each subscription uses a separate database connection.

► Table mappings

After you define a subscription in Management Console, source and target tables can be mapped for replication. Subscriptions can contain as many table mappings as necessary. Management Console provides two mechanisms for mapping:

- Multiple one-to-one table mappings

Map tables using one-to-one replication when you want to map multiple source tables to multiple target tables at a time and these tables share a table structure and similar table names. Using a wizard, Management Console automatically maps tables that are based on an example mapping you set up.

- Custom table mappings

Map tables using custom replication when you want to map only one source table to one target table at a time. Use custom table mappings when you need more control over the way the table is mapped. For example, use custom mappings for tables that do not share a table structure or similar table names, or for mappings where you must filter out columns for a particular table.

► Column mappings

After you establish table mappings, source and target table individual column mapping can be customized using Management Console.

14.3.1 Unidirectional replication

Unidirectional replication is the replication of data changes from source tables to target tables or another destination that is designated by the administrator (Figure 14-3).

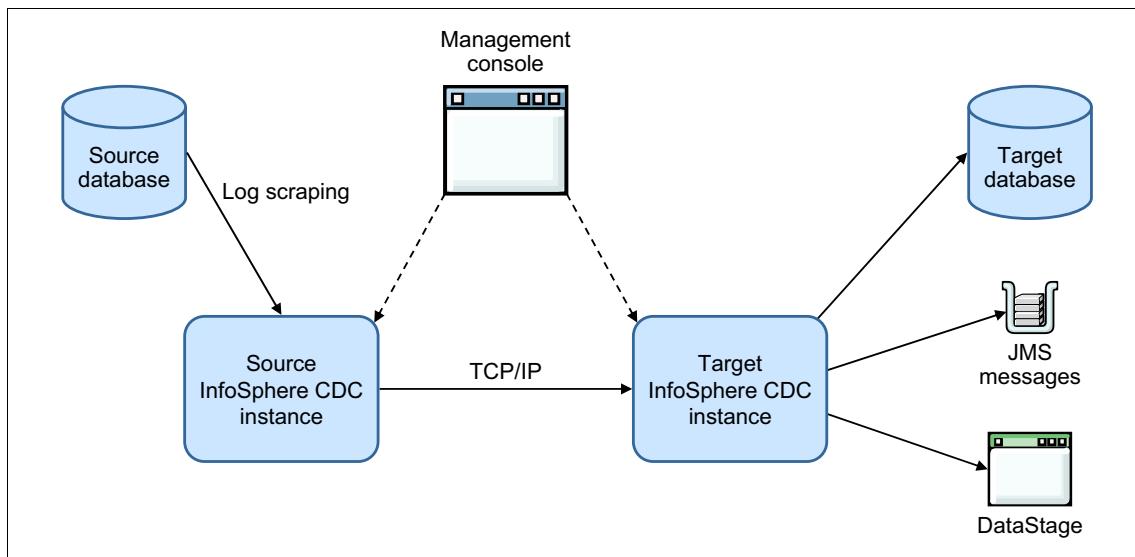


Figure 14-3 Unidirectional replication

After you create a subscription, table mappings control the unidirectional replication scheme that is adopted by InfoSphere CDC. In addition to standard unidirectional replication, there are other types of table mapping setups available with InfoSphere CDC that can tackle particular business needs.

Mapping using standard replication

Standard replication allows target tables to maintain an updated image of the data that is contained in the source table. Source and target tables can be of different types and you can transform the data that you replicate between the source and the target. Under standard replication, InfoSphere CDC applies the same operation that occurred on the source table to the target table. For example, a row update operation on the source table determines a row update operation on the target table.

Mapping using IBM LiveAudit

IBM LiveAudit™ allows your target tables to maintain an audit trail of operations that are applied to the source table. Target tables contain rows that track insert, update, delete, and truncate operations that are applied to the source tables. Target tables may be set to maintain before and after values for source table operations.

Mapping using Adaptive Apply

Adaptive Apply mapping allows error-free replication between source and target tables that are not synchronized. InfoSphere CDC reacts intelligently to operations that take place on the source table. For example, if there is an insert on the source table, but that row exists in your target table, InfoSphere CDC switches the insert to an update operation. Also, if there is an update on your source table, and this row does not exist on your target table, then InfoSphere CDC switches the update into an insert.

14.3.2 Bidirectional replication

Bidirectional replication involves replication from a source data store to a target data store, and replication in the opposite direction (Figure 14-4). Table mappings in both directions may be independent, in which case the setup may be regarded as two independent unidirectional replication schemes. Alternatively, table mappings may be set up to replicate the same table in both directions. This bidirectional table replication scheme calls for mechanisms to prevent recursion, and to detect and resolve conflicts with replicated data.

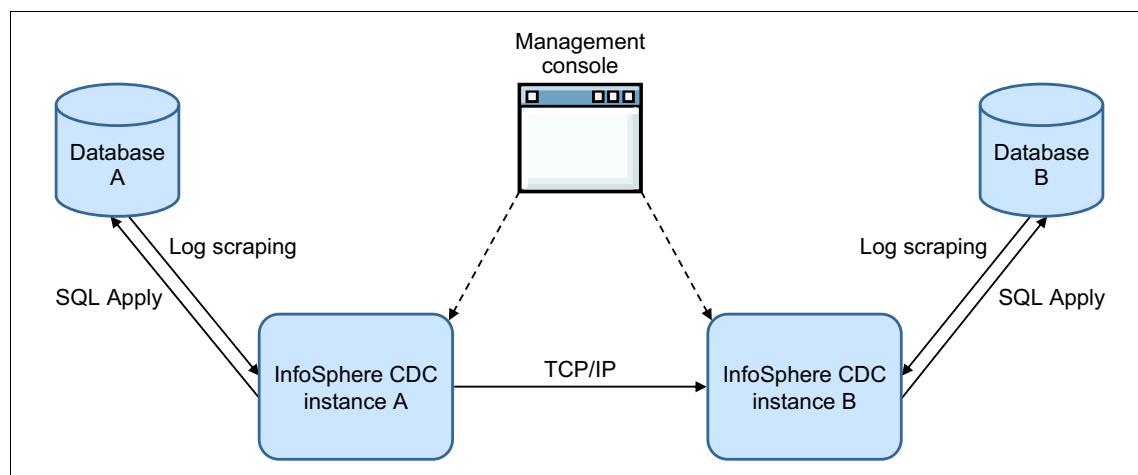


Figure 14-4 Bidirectional replication

If both servers are used to alter the same table data, recursive updates might occur. A change on the source server is replicated to the target server, which may be replicated back to the source. This scenario is prevented by setting Prevent Recursion using the table mapping configuration.

You can use conflict detection and resolution to detect, log, and act on inconsistent data on the target. This ability ensures that the replication environment handles data conflicts automatically and in accordance with business rules. As conflicts are detected and resolved, InfoSphere CDC logs them in a conflict resolution audit table. More information about conflict detection and resolution is presented later in 14.4, “Features and functionality” on page 503.

14.3.3 Replication to other destinations

This section outlines other supported InfoSphere CDC destinations for replication. For more information, see the Information Center at:

<http://publib.boulder.ibm.com/infocenter/cdc/v6r5m1/index.jsp>

JMS message queue

InfoSphere CDC can be set up to send XML messages to a JMS message destination whenever it detects a transaction, boundary, or commit operation on designated tables on the source database.

Flat files

InfoSphere CDC can replicate data changes to flat files, which can then be used to feed an existing ETL solution or data warehouse appliance. Settings include the destination file directory, and when produced files are considered complete (hardened) for subsequent processing. This setting can be based on a particular number of sent rows, or by using a time threshold. To maintain transaction integrity, files can be hardened only on commit boundaries.

InfoSphere DataStage

In addition to using flat files to feed InfoSphere DataStage, the Direct Connect method may be used to allow InfoSphere CDC to connect directly to InfoSphere DataStage over a TCP/IP connection

14.3.4 High availability and disaster recovery with InfoSphere CDC

InfoSphere CDC is an efficient and reliable replication solution that can be used in high availability or disaster recovery systems. In this section, we outline some examples of InfoSphere CDC topologies for that purpose. We then list advantages and limitations of InfoSphere CDC when used in a high availability or disaster recovery scheme.

- ▶ Unidirectional replication to a disaster recovery site

InfoSphere CDC can be used to establish unidirectional replication between a primary production server and a secondary server.

- ▶ Bidirectional replication to establish an active-active setup

InfoSphere CDC can be used to maintain synchronization between two disparate active servers. InfoSphere CDC recursion prevention, and its conflict detection and resolution capabilities, maintain data integrity and consistency across both servers.

InfoSphere CDC has the following advantages:

- ▶ Replication can be set up to take place continuously in real time or periodically using batches, for example, using hourly or nightly batches. More information about types of replication is presented in 14.4, “Features and functionality” on page 503.
- ▶ Platforms, operating systems, and database servers (or versions) for sources and targets of replication do not need to be the same. This situation is suitable for disaster recovery sites that are intended for active outdated production servers. Furthermore, this situation allows for lower-cost hardware to be used for secondary sites.
- ▶ InfoSphere CDC instances may be installed on the server alongside the source or target database, or on a separate server that connects remotely to the database server. Although InfoSphere CDC employs a non-invasive approach to extracting changed data from production servers, having the instance on a separate server can avoid resource consumption by InfoSphere CDC on the production server.
- ▶ InfoSphere CDC can be configured to replicate specific tables, or specific columns within specific tables. Replication of only sensitive or critical data might suffice in some business cases.
- ▶ There is no distance limitation between replicated sources and targets with InfoSphere CDC.

InfoSphere CDC has the following limitations:

- ▶ Non-logged operations are not mirrored. Examples include LOAD operations on source tables. With such operations, refresh mode synchronization is required to propagate changes.
- ▶ DB2 Data Description Language (DDL) changes on the source data store are not captured and are not propagated to the target data store.
- ▶ Encrypted data can be replicated only between similar systems and databases. For example, an encrypted database on a DB2 for Linux, UNIX, and Windows server can be replicated only to a similar DB2 for Linux, UNIX, and Windows server.

14.4 Features and functionality

In this section, we present some of the various features and functionalities available with InfoSphere CDC.

14.4.1 Transformations

Transformations are needed when data is shared between heterogeneous data stores. InfoSphere CDC incorporates advanced and diverse transformation capabilities, some of which are:

- ▶ Data translations on column mappings

You can use Management Console to translate specific data in source columns to new data before mapping to target columns. For example, you may have a source column that is called CITY containing entries like NY, TO, and CAI, which are codes that you want translated to a target column with their full names (New York, Toronto, and Cairo). This ability is useful if you have few data values or symbols that must be mapped to different values. It is not practical with large data sets.

- ▶ Column functions

InfoSphere CDC provides a number of column functions that you can use in expressions when you are mapping target columns. This capability is suitable when low-level data manipulation is needed in the replication flow.

The following categories outline the different sets of functions available with InfoSphere CDC:

- String functions

During replication, you can have InfoSphere CDC concatenate multiple strings, remove blank characters from a string, change the case of the characters in a string, replace characters of a string with other characters, or extract a substring from a string.

- Date and time functions

InfoSphere CDC can manipulate date and time values during replication. You can add a two-digit century specification to a date, or retrieve the current date and time.

- Conversion functions

InfoSphere CDC can convert values from one data type to another during replication. You can convert from any value to a character string or a numeric value, or from a numeric or character value to a datetime-type value.

- Conditional and variable functions

InfoSphere CDC can evaluate an expression and return different results (IF-ELSE behavior). InfoSphere CDC can also declare a new variable, assign a value to it, or retrieve the value of an existing variable.

- Data functions

InfoSphere CDC can provide before or current values for rows. This capability is useful if decisions must be made based on previous values for columns.

User exits

In addition to the translation and transformation functions that are described in the previous section, the user exit feature can be used to allow external programs to manipulate data. External functions may be stored procedures or Java programs. User exits allow InfoSphere CDC to start the defined subroutines when a predefined event occurs.

Two types of user exits are available:

- ▶ Table-level user exits run a set of actions before or after a database event occurs on a specified table.
- ▶ Subscription-level user exits run a set of actions before or after a commit event occurs on a specified subscription.

You can use a table-level user exit to define a set of actions that InfoSphere CDC can run before or after a database event occurs on a specified table. A database event is defined as either a row-level operation or as a table-level operation.

Row-level operations include an insert, update, or a delete. Table-level operations include a refresh or a truncate operation. For example, you can configure a row-level user exit program that sends an alert after InfoSphere CDC replicates a delete operation on a particular target table.

You can use a subscription-level user exit to define a set of actions that InfoSphere CDC can run before or after a commit event occurs on a specified subscription.

14.4.2 Replication modes

After you create subscriptions and mapping on your tables between the source and target data stores, you can configure replication using the Management Console. InfoSphere CDC supports two modes of replication:

- ▶ Refresh
Refresh replication usually performs the initial synchronization of the tables from the source database to the target.
- ▶ Mirror
Mirroring handles the continuous replication of ongoing changes that take place at the source data store.

Refresh mode replication

The InfoSphere CDC refresh operation is designed to synchronize source and target tables. Source table data is read by the Refresh Reader at the source InfoSphere CDC instance. With replication set up and in operation, tables might need to be brought out of synchronization for a configuration change or for maintenance operations.

Suspending replication for a particular table is referred to as *parking*. For example, parking a table can be done to update the definition of the source table. Changes that are taking place on the source during parking are not replicated to the target.

After configuration changes or maintenance operations are complete, parked tables can be flagged for a refresh. Tables that are flagged for refresh and have a replication method of Mirror can be refreshed before mirroring begins. InfoSphere CDC refreshes all of the flagged tables within a single subscription as one sequential operation that runs to completion. Each table is refreshed individually, one at a time, until all flagged tables are finished refreshing.

InfoSphere CDC offers two types of refresh operations:

- ▶ Standard refresh

A standard refresh results in a complete copy of the data in a source table being sent to the target table. This operation truncates or deletes the contents of the target table and then inserts the new contents that are sent by the source system.

- ▶ Differential refresh

A differential refresh updates the target table by applying only the differences between it and the source table. Instead of the target table being cleared at the start of the refresh and repopulated with data, differential refresh compares each row in the target table with each row in the source table to identify missing, changed, or additional rows. The primary advantage of a differential refresh is that the target table stays online during the refresh operation. A differential refresh can also track all performed changes on the target during the refresh in a log table that is created in the target replication engine. Furthermore, a differential refresh can log the necessary changes without actually performing them on the target.

With tables that have referential integrity constraints, you can set a refresh order to preserve these constraints during a refresh. For example, you might want InfoSphere CDC to refresh your DEPARTMENT tables before you refresh your EMPLOYEE tables, based on the fact that each employee belongs to a specific department. You can change the order in which InfoSphere CDC refreshes your table mappings by moving tables into groups. Each table that you decide to move in to a group is assigned a sequence number that InfoSphere CDC uses to refresh each table mapping in numerical order. Any remaining table mappings that you did not add to a group are refreshed in an arbitrary order by InfoSphere CDC after all groups are refreshed.

Mirror mode replication

InfoSphere CDC mirroring uses change data capture replication through log scraping. InfoSphere CDC provides two types of mirroring: Continuous and Scheduled End (Net Change):

- ▶ Continuous mirroring

Continuous mirroring replicates changes to the target on a continuous basis. Use this type of mirroring when business requirements dictate that you need replication to be running continuously with the source and target synchronized always and you do not have a clearly defined point-in-time to end replication. Continuous mirroring is also useful if your environment experiences frequent changes to large volumes of data. Instead of capturing these changes using a batch transfer, you can replicate changes to the target on a continuous basis.

- ▶ Scheduled End (Net Change) mirroring

Scheduled End (Net Change) mirroring replicates changes (to the target) up to a user-specified point in the source database log and then ends replication. Use this type of mirroring when business requirements dictate that you replicate data only periodically and you have a clearly defined end point for the state of your target database when replication ends. You can use Scheduled End (Net Change) mirroring to end replication at the following points in your source database log:

- Current time or “Now”
- User-specified date and time
- User-specified log position

These user specified end points ensure that your target database is in a known state when replication ends.

14.4.3 Filtering

InfoSphere CDC replication can be set up to filter particular rows or columns for replication. Furthermore, particular “critical” columns can be set to control whether rows are replicated when changes take place on the source table.

- ▶ Filtering rows

To include or exclude particular rows for replication, you must build a row-filtering expression. All row-filtering expressions that you define must return a Boolean result. For example, you might have a source column such as SALARY that maintains the salary for each employee in your organization. You might want to replicate only those rows to the target table for those employees that have a salary greater than \$48,000. In this scenario, you must define a row-filtering expression (`SALARY > 48000`). You can use column manipulation functions, basic numeric operators, and **SQL SELECT WHERE** clauses in your row-filtering expressions.

- ▶ Critical columns

By default, InfoSphere CDC replicates inserts, updates, and deletes to the target table during replication. However, you can control the updates that InfoSphere CDC replicates using the select critical column feature. When you select a column as critical, InfoSphere CDC compares the before and after image of the row to determine if the critical column value changed during an update. A row is only replicated for update operations when a critical column has a value that differs from the before image.

For example, you might have a source table that maintains customer account information. Instead of receiving every update that is made to the source table, you might want only the target table to receive the row when the customer account balance is updated. In this scenario, you mark the customer account balance column as a critical column. InfoSphere CDC replicates only this row when there are updates that are made to that column that result in a changed value. If the column is updated, but the value stays the same, the row cannot be replicated.

- ▶ **Filtering columns**

By default, InfoSphere CDC replicates all mapped source columns to the target table. If there is a source column you want to exclude for replication, then you can filter it out using Management Console. Excluding source columns for replication might become necessary if the column contains confidential information that you do not want the target to receive.

14.4.4 Conflict detection and resolution

You can use conflict detection and resolution to detect, log, and act on inconsistent data on the target. This capability ensures that your replication environment handles data conflicts automatically and in accordance with your business rules. This situation is important with bidirectional replication schemes. As conflicts are detected and resolved, InfoSphere CDC logs them in a conflict resolution audit table. Conflict resolution is enabled from Management Console and can be set only for individual columns.

During replication, InfoSphere CDC detects conflicts when you:

- ▶ Insert a row and the row's key already exists in the target table. This situation violates the unique key constraint.
- ▶ Update a row and the row's key does not exist in the target table.
- ▶ Update a row and the contents of the rows in the source table and target table, before the update, do not match.
- ▶ Delete a row and the row's key does not exist in the target table.
- ▶ Delete a row and the contents of the rows in the source table and target table, before the delete, do not match.

Conflict resolution is set using Management Console. Whenever a conflict is detected, the following conflict resolution outcomes can be set to take place:

- ▶ **Source wins**

When InfoSphere CDC resolves conflicts so that the source column wins, it applies the row from the source table to the target table. This ensures that the target table row matches the data in your source table upon replication.

- ▶ Target wins

When InfoSphere CDC resolves conflicts for target wins, it does not apply any source changes to the target table. This preserves the row in the target table, as InfoSphere CDC does not apply data from the source in the event of a conflict.

- ▶ Largest value wins

When InfoSphere CDC resolves conflicts for largest value wins, it applies the change to the target if the source row has a larger value than the corresponding row on the target table. InfoSphere CDC treats NULL values as the smallest possible value. If the row does not exist on the target table, then InfoSphere CDC uses NULL as the comparison value. If InfoSphere CDC detects the conflict while it is deleting a row, then it uses the before image of the source table and compares it to the target value.

- ▶ Smallest value wins

When InfoSphere CDC resolves conflicts for smallest value wins, it applies only the change to the target if the value in the source row is smaller than the corresponding row on the target table. Like the largest value wins resolution, InfoSphere CDC treats NULL values as the smallest possible value. If the row does not exist on the target table, then InfoSphere CDC uses NULL as the comparison value. If InfoSphere CDC detects the conflict while it is deleting a row, then it uses the before image of the source table and compares it to the target value.

- ▶ User exit programs

When InfoSphere CDC resolves conflicts with a user exit program, it applies the value that is returned by the user exit program to the target table to resolve the conflict. For example, if a conflict occurs between numerical quantity columns, you can create a user exit program that returns the sum of both source and target values for the target.



Geographically dispersed high availability and disaster recovery solutions

In this chapter, we concentrate on high availability (HA) scenarios with IBM PowerHA SystemMirror software, where nodes are in geographically dispersed locations.

One of the objectives in achieving high availability is to prevent a site from becoming a single point of failure (SPOF). Using geographically dispersed sites eliminates this possibility, because you have a standby site some distance from the primary site.

We describe both storage area network (SAN) SAN Volume Controller mirroring and Geographical Logical Volume Manager (GLVM) mirroring. We show steps to configure simple configuration of GLVM with local and remote disks and show steps to configure and implement GLVM with PowerHA and GLVM.

IBM DB2 pureScale Geographically Dispersed pureScale Cluster (GDPC) is not described here. For more information about GDPC, see:

<http://www.ibm.com/developerworks/data/library/long/dm-1104purescalegdp.html>

This chapter covers the following topics:

- ▶ PowerHA over extended distances
- ▶ PowerHA data replication components
- ▶ Configuring a stand-alone GLVM
- ▶ Manual failover
- ▶ Configuring PowerHA with GLVM

15.1 PowerHA over extended distances

A single server with one copy of your corporate data is single point of failure. To improve the availability of mission-critical data and increase data protection, you must eliminate single points of failure. Several strategies are available to customers to provide access to data with different levels of data protection against accidental loss of important information.

One strategy is adding a second volume group with mirrored disks, which protects against disk failure or disk adapter failure (Figure 15-1). Critical data is duplicated (mirrored) from a principal volume group to a copy of the principal volume group (VG) (mirrored volume group). This configuration uses AIX Logical Volume Manager (LVM) mirroring.

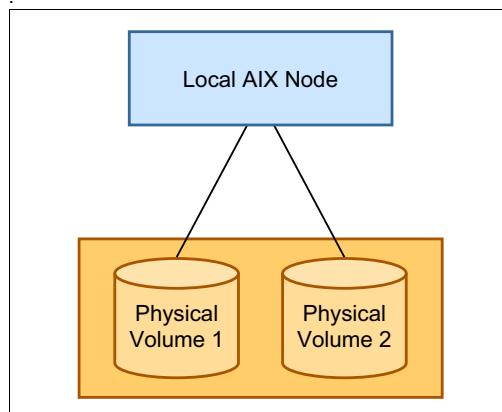


Figure 15-1 Single node with single VG

Mirroring a volume group eliminates a disk (and disk adapter) single point of failure (Figure 15-2). However, the server is still a potential single point of failure on the node.

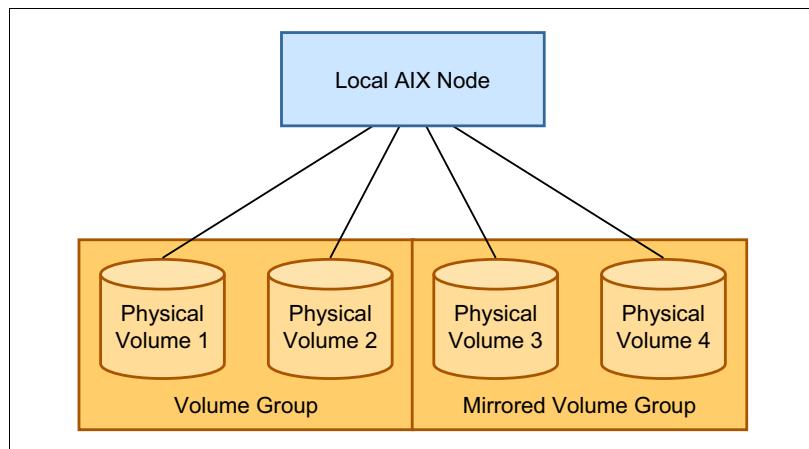


Figure 15-2 Single node with mirrored VG

PowerHA enables a second server to be part of the cluster with the primary server accessing a mirrored VG and a standby server that can take over the function of the primary server (failover) if the primary server experiences a failure (Figure 15-3).

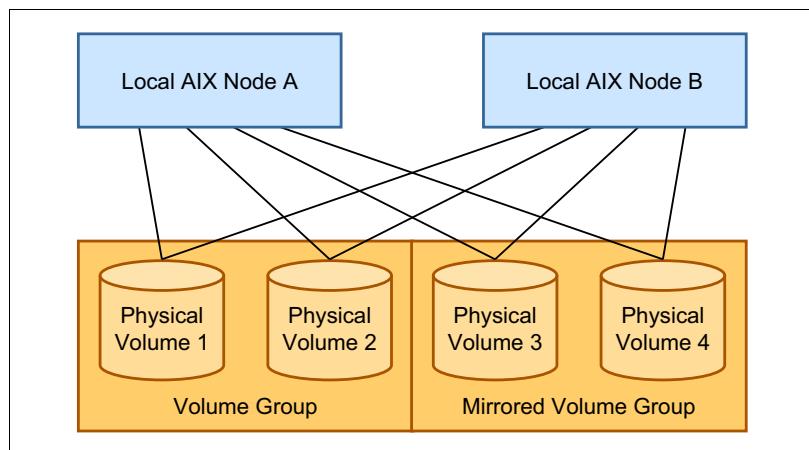


Figure 15-3 PowerHA node with mirrored VG

Both the disk and disk adapter single points of failure can be eliminated by the AIX LVM mirrored service. A server single point of failure can be eliminated by using PowerHA cluster technology.

There is another single point of failure, that is, the site. A disaster, such as a fire, explosion, a catastrophic power failure, at the primary site can possibly destroy all the servers at the site. If both the primary and standby servers are at the same site, all your data could be lost.

You can use PowerHA with GLVM to place servers with mission-critical data in geographically dispersed sites that are several kilometers apart and use separate power grids. Two sites, one called the *primary site* and the other called the *remote site*, each host servers, that is, the primary server and remote server (Figure 15-4). Mission-critical data is on a mirrored VG that spans both sites. Remote disk access is enabled through a Remote Physical Volume (RPV) device driver and a corporate Internet Protocol network.

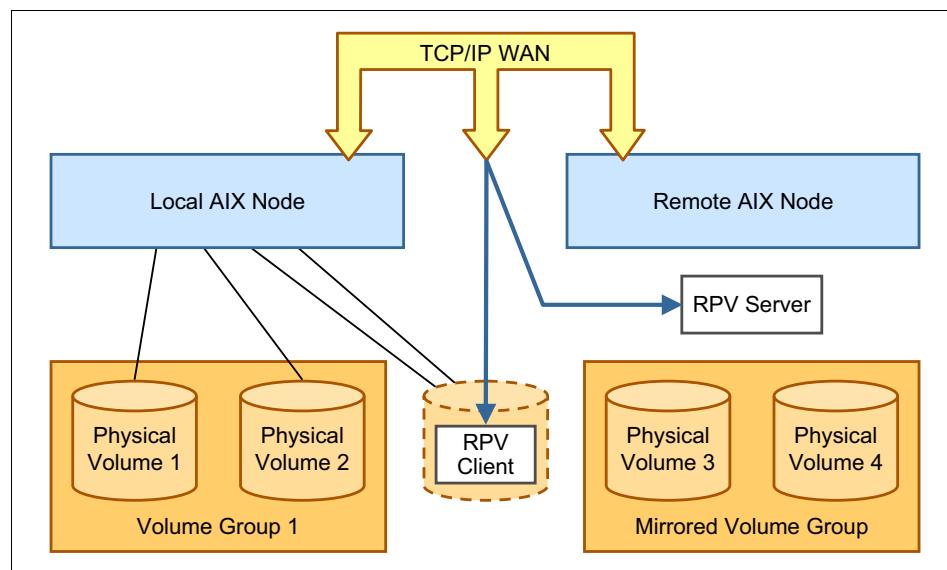


Figure 15-4 PowerHA/GLVM complex

All single points of failure can be eliminated, except for the network itself.

PowerHA with GLVM allows up to four network connections between the primary and remote sites. Those network connections are used for performance reasons, but they also add redundancy for remote disk access.

You can use the PowerHA software to ensure continuous operation by enabling a recovery of failed component within a single node at a single site by using a primary and standby server.

PowerHA uses this basic capability of PowerHA and allows fallback and failover between geographically dispersed nodes.

GLVM can be configured as a stand-alone solution or it can be combined with PowerHA. When GLVM is configured as a stand-alone solution, the application can access the same data regardless of where the application runs, whether on the primary site or on the remote site. The application cannot run on both sites at the same time.

PowerHA with GLVM manages the failback and failover between primary and remote sites, automates the failover and failback processes, adds automatic failure detection and manages resources, for example, the GLVM mirrored volume group.

The main function of the PowerHA software is to manage data replication between local and remote nodes in a manner that is transparent to the application.

15.2 PowerHA data replication components

In this section, we describe various components that can be used to build a geographically dispersed solution based on PowerHA and GLVM.

15.2.1 PowerHA with SAN Volume Controller mirroring

PowerHA takes advantage of SAN advanced features, such as SAN Volume Controller mirroring capabilities.

Data is replicated using the SAN Volume Controller data copy services capability of the SAN subsystem. Copy services protect a mirrored copy from accidental modification and ensure mirroring.

The PowerHA software provides failover and fallback capabilities and ensures the integrity of the replicated data.

15.2.2 PowerHA with Geographical Logical Volume Manager

PowerHA/XD with GLVM enables data mirroring and disaster recovery capabilities for geographically dispersed sites. Remote mirroring protects data from being lost because of a total failure of one site. Remote mirroring over WAN does not impose any distance limitation except for data propagation delays that depend directly on link speeds. Remote mirroring enables access to data that is replicated from both sites, thus enabling a business application to continue running at a remote site while the local site is being repaired or serviced.

PowerHA automates the process of failover and fallback between local and remote nodes and allows the inclusion of mirrored volume groups that consist of a physical volume (PV) and remote physical volume (RPV) in the PowerHA cluster.

PowerHA with GLVM has two main functions: remote data mirroring and automated failover and fallback. Remote data mirroring is achieved by using an RPV

RPV components

An RPV has two components:

- ▶ RPV client
- ▶ RPV server

An RPV client acts as a device driver, which appears to a source node as a disk drive, but reroutes all I/O operation over WAN to the RPV server. It runs on the node where the application issues I/O requests to the physical volumes.

An RPV server runs on the target node and completes I/O to the physical volume that is replicated from the source node. It acts as a kernel extension and not as an actual physical device.

For mutual failover and fallback, it is necessary to configure both pairs of the RPV client and the PV server, with one pair for each site.

An RPV client is configured on the primary site and completes I/O on the remote site by communicating with the RPV server at the remote site. Geographic mirroring occurs between the RPV client and the RPV server. To enable takeover in either direction, an RPV client that is configured at the remote site must complete I/O on the primary site.

15.2.3 Geographical Logical Volume Manager

GLVM is based on Logical Volume Manager (LVM) concepts and extends the mirroring capabilities of LVM to remote node by using an RPV client to connect to an RPV server over a WAN.

LVM data mirroring performs write operations to a primary physical disk and to a secondary disk (both disks are local) before it returns a notification of success about the write operation. This classical mirroring technique prevents a single point of failure when there is a local disk failure.

GLVM data mirroring performs write operation to a primary disk and to a secondary disk that is implemented as an RPV client that sends data to be written to a remote RPV server, which completes a write operation to the remote server's physical disk.

GLVM is implemented as a symmetric pair (source to remote and remote to source) mirroring write operation, thus enabling failover and fallback.

GLVM is not aware that the disk is participating in the mirrored write operation is at the remote node. During disk write operation, both the RPV client device driver and LVM perform a single logical write I/O as a pair of physical writes to the local PV and remote RPV; both the PV and RPV belong to the same VG.

During read I/O, LVM attempts to complete an operation from the local PV rather than the remote RPV to enhance I/O performance.

A remote RPV can be thought of as an RPV client device driver connected over WAN with the RPV server; this pair of device drivers implements a physical remote I/O operation.

A simple local node and remote node cluster with mirrored volumes can be further extended to become a multinode local cluster with fallback and failover capabilities to a remote multinode cluster.

To prevent site partitioning and data divergence, in addition to the network that carries I/O requests to remote site, you must configure a separate network that sends heartbeats. A heartbeat network prevents node partitioning (split-brain syndrome) if there is a catastrophic failure, where all the I/O request networks fail.

15.2.4 Synchronous and asynchronous data mirroring

GLVM can be configured for synchronous and asynchronous mirroring. Synchronous versus asynchronous choices are dictated by network bandwidth and network latency.

In synchronous mode, each logical volume write I/O is considered complete when data is written to both the local and remote mirrors. This mode of operation ensures that both sites have a current copy of the data. The advantage of using synchronous mode is that logical volume read I/O is faster, because data is on a local disk; there is no data loss when either node fails. A limitation of synchronous mode is that logical write I/O that depends on network throughput is slower. Logical write operations now depend on both network latency and network bandwidth.

Network saturation happens when data sent over the WAN exceeds the network's capacity. If the network operates at capacity, a slowdown in writing transmitted buffers occurs. This saturation can be remedied by either migrating to a higher bandwidth network, by adding another network link, or by applying both strategies. The RPV device driver supports up to four data networks that carry I/O requests.

Network latency depends mainly on the distance between the RPV client and RPV server; the speed of the network determines latency. For each I/O request, the time for a round trip of the I/O request and the corresponding I/O status effectively limits the distance over which using LVM in synchronous mode is feasible.

When an application workload and the corresponding requirements for I/O completion are such that the network latency exceeds those requirements, asynchronous mirroring is an alternative option. In the asynchronous mode of data mirroring, the primary site I/O requests complete while the remote site I/O requests (when you consider primary to remote mirroring) are updated asynchronously, that is, they are buffered, cached, and transmitted as the network bandwidth allows. The advantage of asynchronous mode is that logical write I/O operations depend on local disk write time, thus improving application response time. Bursts of intense write operations are absorbed by the local cache. A limitation of this mode is that there is some risk of data loss, because the remote node has backed up data. There is also possibility of data divergence.

In addition, when the workload of an application is so large that the local buffers and the cache become full, there is no advantage to using asynchronous mirroring. So, the only approach is to use synchronous mirroring with an increase in network bandwidth or the speed or number of networks that are used in data mirroring.

Your decision whether to use GLVM or HADR depends on the DB2 workload and the number of databases. HADR defines a HA relationship between a pair of databases and achieves its goal by using transaction logs and transaction log buffers replication. The amount of data that is transferred over the WAN depends on the transaction log file size, which is controlled by the database configuration parameter **LOGFILSIZ**, and on the transaction log buffer size, which is controlled by the database configuration parameter **LOGBUFSIZ**, for each of the databases that participate in the HADR setup. GLVM mirrors disks, and the amount of data that is transferred over the WAN depends on the number of write I/O operations that are performed on file systems on the mirrored disks.

For a single pair of databases, HADR might be more efficient compared to GLVM. For workloads that are not write intensive, GLVM might be a more satisfactory solution. Read I/O operations are completed using a local copy of the data rather than a mirrored remote copy; the read I/O is less dependent on the throughput of the WAN link. Write I/O operation, in synchronous mode, must be complete on the local and remote copies for each write operation, and are more dependent on WAN link throughput.

15.3 Configuring a stand-alone GLVM

GLVM is part of the PowerHA Enterprise Edition. The following examples are based on PowerHA Enterprise Edition in PowerHA 6.1 EE for AIX V6.1.

Our test environment consists of two nodes:

- ▶ NC047089.kraklab.p1.ibm.com (*localnode*)
- ▶ NC047094.kraklab.p1.ibm.com (*remotenode*)

NC047089 (*localnode*) has a public IP address of 9.156.47.89 and a private IP address of 10.1.1.14 (which is used for GLVM data traffic).

NC047094 (*remotenode*) has a public IP address of 9.156.47.94 and a private IP address of 10.1.1.16 (which is used for GLVM data traffic).

Our sample starting disk configuration is shown in Example 15-1.

Example 15-1 Sample disk configuration

```
[root@NC047089.kraklab.p1.ibm.com] : # lspv
hdisk0          00cca5e4a4fd37c2                  rootvg
active
hdisk1          00cca5e4b9b5ffe1                  None
hdisk2          00cca5e4ffa033b7                  None

[root@NC047094.kraklab.p1.ibm.com] : # lspv
hdisk0          00cca5e4a4fe0209                  rootvg
active
hdisk1          00cca5e4b9b5ffe1                  None
hdisk2          00cca5e4ff9dd0cb                  None
```

Using physical volume identifiers (PVID), hdisk1 is a shared disk between the two nodes. hdisk1 is not used in the GLVM stand-alone setup. hdisk2 is not shared between both nodes.

To configure GLVM, complete the following steps:

1. Install the GLVM related file sets.

From the PowerHA Enterprise Edition distribution media, install the following file sets:

- glvm.rpv.client 6.1.0.0 COMMITTED Remote Physical Volume Client
- glvm.rpv.server 6.1.0.0 COMMITTED Remote Physical Volume Server
- glvm.rpv.util 6.1.0.0 COMMITTED Geographic LVM Utilities

You can install these file sets by using either **smit** (GUI) or **smitty** (CLI). The **smitty glvm_utils** command opens the main menu from which most of the configuration steps be run, except those steps that use the base AIX LVM configuration. All configuration steps are performed as the root user.

2. Define node names.

Run **smitty glvm_utils** and select Remote Physical Volume Servers → Remote Physical Volume Server Site Name Configuration. Name the nodes as follows:

- Node NC047089 is named localnode.
- Node NC047094 is named remotenode.

3. Define an RPV server on remotenode.

On remotenode, run **smitty glvm_utils** and select Remote Physical Volume Servers → Servers → Add Remote Physical Volume Server. Complete the following steps:

- a. Select hdisk2 from the list of available disks.
- b. Select 10.1.1.14 as the Remote Physical Volume Client IP address.
- c. Set Configure Automatically at System Restart and Start New Devices Immediately options are set to no.

Upon successful execution, rpvserver0 is created in a defined state (Example 15-2).

Example 15-2 Remote Physical Volume Server configuration

Change / Show a Remote Physical Volume Server

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

Fields	[Entry]
Remote Physical Volume Server	rpvserver0
Physical Volume Identifier	
00cca5e4ff9dd0cb	
Remote Physical Volume Client IP address	[10.1.1.14]
+	
Configure Automatically at System Restart?	[no]

4. Define an RPV client on localnode.

To define an RPV client on localnode, an RPV server on remotenode must be in the available state. Complete the following steps:

- On remotenode, run **smitty glvm_utils** and select Remote Physical Volume Servers → Configure Defined Remote Physical Volume Servers.
- Select rpvserver0 and put it in to the available state.

To verify the state of the RPV server, run **lsdev -t rpvstype** (Example 15-3).

Example 15-3 Verify RPV server state with rpvstype command

```
[root@NC047094.kraklab.pl.ibm.com] :/ # lsdev -t rpvstype
rpvserver0 Available  Remote Physical Volume Server
```

- On localnode, run **smitty glvm_utils** and select Remote Physical Volume Clients → Add Remote Volume Clients. Complete the following steps:
 - For Remote Physical Volume Server IP address, enter a private network address of 10.1.1.16 for remotenode.
 - For Remote Physical Volume Local IP address, enter a private network address of 10.1.1.14 for localnode.
 - Set Start New Devices Immediately to no.

Upon successful completion of the command, a “virtual” disk named hdisk3 is created and put in the defined state. The **lsdev -t rpvcclient** command shows the RPV client state and virtual disk name. The **lsattr -El hdisk3** command shows all the details of the defined virtual disk (Example 15-4).

Example 15-4 RPV virtual disks details

```
[root@NC047089.kraklab.pl.ibm.com]#/ # lsdev -t rpvcclient  
hdisk3 Defined Remote Physical Volume Client  
  
[root@NC047089.kraklab.pl.ibm.com]#/ # lsattr -El hdisk3  
io_timeout 180 I/O Timeout Interval  
True  
local_addr 10.1.1.14 Local IP Address  
(Network 1) True  
local_addr2 none Local IP Address  
(Network 2) True  
local_addr3 none Local IP Address  
(Network 3) True  
local_addr4 none Local IP Address  
(Network 4) True  
pvid 00cca5e4ff9dd0cb0000000000000000 Physical Volume  
Identifier True  
server_addr 10.1.1.16 Server IP Address  
(Network 1) True  
server_addr2 none Server IP Address  
(Network 2) True  
server_addr3 none Server IP Address  
(Network 3) True  
server_addr4 none Server IP Address  
(Network 4) True
```

You created an RPV client to RPV server definition from localnode to remotenode. Now you must set up the RPV client to RPV server definition remotenode to localnode. Complete the following steps:

- a. On remotenode, run **smitty glvm_utils** and select Remote Physical Volume Servers → Remove Remote Physical Servers
- b. Select rpserver0 set Keep definition in database to yes.

Upon successful completion of this command, the RPV server rpserver0 is in the **defined** state (Example 15-5).

Example 15-5 Verify RPV server defined - checking

```
[root@NC047094.kraklab.pl.ibm.com] :/ # lsdev -t rpvstype  
rpvserver0 Defined Remote Physical Volume Server
```

- c. On localnode, run **smitty glvm_utils** and select Remote Physical Volume Servers → Servers → Add Remote Physical Volume Server:
 - i. Select hdisk2 from the list of available disks.
 - ii. Select 10.1.1.16 for Remote Physical Volume Client IP address.
 - iii. Set Configure Automatically at System Restart and Start New Devices Immediately to no.

Upon successful execution, rpvserver0 is created in the defined state.

To define an RPV client on remotenode, an RPV server on localnode must be in the available state. Complete the following steps:

- a. On localnode, run **smitty glvm_utils** → Remote Physical Volume Servers → Configure Defined Remote Physical Volume Servers.
- b. Select rpserver0. Upon successful completion of the command, it is in the available state.

You can run **lsdev -t rpvstype** to verify the state of the RPV server (Example 15-6).

Example 15-6 lsdev -t rpvstype used to verify the state of the RPV server

```
[root@NC047089.kraklab.pl.ibm.com] :/ # lsdev -t rpvstype  
rpvserver0 Available Remote Physical Volume Server
```

- c. On remotenode, run **smitty glvm_utils** and select Remote Physical Volume Clients → Add Remote Volume Clients:
 - i. For Remote Physical Volume Server IP address, enter a private network address of 10.1.1.14 for localnode.
 - ii. For Remote Physical Volume Local IP address, enter a private network address of 10.1.1.16 for remotenode.
 - iii. Set Start New Devices Immediately to no.

Upon successful completion of the command, a virtual disk named hdisk3 is created and is in the defined state.

The **lsdev -t rpvclient** command shows the RPV client state and virtual disk name. The **lsattr -El hdisk3** command shows all the details of the defined virtual disk.

The following states should exist for the RPV clients and servers, depending on the direction of replication:

- localnode to remotenode:
 - On localnode, the RPV client is available and the RPV server is defined.
 - On remotenode, the RPV client is defined and the RPV server is available (Example 15-7).

Example 15-7 lsdev command output

```
[root@NC047089.kraklab.pl.ibm.com]:# lsdev -t rpvclient  
hdisk3 Available Remote Physical Volume Client  
[root@NC047089.kraklab.pl.ibm.com]:# lsdev -t rpvtstype  
rpvserver0 Defined Remote Physical Volume Server  
  
[root@NC047094.kraklab.pl.ibm.com]:# lsdev -t rpvclient  
hdisk3 Defined Remote Physical Volume Client  
[root@NC047094.kraklab.pl.ibm.com]:# lsdev -t rpvtstype  
rpvserver0 Available Remote Physical Volume Server
```

- remotenode to localnode:
 - On localnode, the RPV client is defined and the RPV server is available.
 - On remotenode, the RPV client is available and the RPV server is defined.

5. Create volume groups.

To create a volume group, run a SMIT command. On localnode, run the following command:

```
smitty mkvg
```

A scalable volume group named **glvm_vg** is created. Both the local disk **hdisk2** and virtual disk **hdisk3** are selected. The option **Activate volume group AUTOMATICALLY at system restart** is set to no.

Volume Group MAJOR NUMBER is set to the first available number in both localnode and remote node. It can be verified by running **lvolstmajor** on both nodes. You should pick one common number as a preferred practice (Example 15-8).

Example 15-8 Creating volume groups

```
* VOLUME GROUP name                               glvm_vg
* Activate volume group AUTOMATICALLY          no
+
    at system restart?
* A QUORUM of disks required to keep the volume   no
+
    group on-line ?
    Convert this VG to Concurrent Capable?      no
+
    Change to big VG format?                     no
+
    Change to scalable VG format?                no
+
    LTG Size in kbytes                         256
+
    Set hotspare characteristics               n
+
    Set synchronization characteristics of stale  n
+
    partitions
        Max PPs per VG in units of 1024       32
+
        Max Logical Volumes                   256
+
        Mirror Pool Strictness
```

QUORUM of disks required should be set to no. You can change this value by running the following command:

```
chvg -Q n glvm_vg
```

You can also change it by running **smitty chvg**.

After creating the glvm_vg volume group, vary it online by running the following command:

```
varyonvg glvm_vg
```

6. Create logical volumes.

Create two logical volumes in glvm_vg volume group by running **smitty mklv**:

- glvm_lv (JFS2 logical volume).
 - glvm_log (JFS2 log logical volume).
- a. Set PHYSICAL VOLUME names to local disk hdisk2 only.
 - b. Set Number of COPIES of each logical partition to 1.
 - c. Set Allocate each logical partition copy on a SEPARATE physical volume to superstrict (Example 15-9).

Example 15-9 Creating logical volumes

```
LOGICAL VOLUME: glvm_lv VOLUME GROUP:  
glvm_vg  
LV IDENTIFIER: 00cca5e400004c000000013905551887.1  
PERMISSION: read/write  
VG STATE: active/complete LV STATE:  
opened/syncd  
TYPE: jfs2 WRITE VERIFY: off  
MAX LPs: 512 PP SIZE: 8  
megabyte(s)  
COPIES: 2 SCHED POLICY:  
parallel  
LPs: 256 PPs: 512  
STALE PPs: 0 BB POLICY:  
relocatable  
INTER-POLICY: minimum RELOCATABLE: yes  
INTRA-POLICY: middle UPPER BOUND: 1  
MOUNT POINT: /glvm_fs LABEL:  
/glvm_fs  
MIRROR WRITE CONSISTENCY: on/ACTIVE  
EACH LP COPY ON A SEPARATE PV ?: yes (superstrict)  
Serialize IO ?: NO  
DEVICE SUBTYPE : DS_LVZ  
COPY 1 MIRROR POOL: None  
COPY 2 MIRROR POOL: None  
COPY 3 MIRROR POOL: None  
  
LOGICAL VOLUME: glvm_log VOLUME GROUP:  
glvm_vg  
LV IDENTIFIER: 00cca5e400004c000000013905551887.2  
PERMISSION: read/write  
VG STATE: active/complete LV STATE:  
opened/syncd
```

```

TYPE:          jfs2log           WRITE VERIFY:  off
MAX LPs:       512              PP SIZE:      8
megabyte(s)
COPIES:        2                SCHED POLICY:
parallel
LPs:          256              PPs:          512
STALE PPs:    0                BB POLICY:
relocatable
INTER-POLICY: minimum          RELOCATABLE: yes
INTRA-POLICY: middle           UPPER BOUND: 1
MOUNT POINT:  N/A              LABEL:        None
MIRROR WRITE CONSISTENCY: on/ACTIVE
EACH LP COPY ON A SEPARATE PV ?: yes (superstrict)
Serialize IO ?: NO
DEVICE SUBTYPE : DS_LVZ
COPY 1 MIRROR POOL: None
COPY 2 MIRROR POOL: None
COPY 3 MIRROR POOL: None

```

The JFS2 log logical volume is initialized by running the following command:

```
logform /dev/glvm_log
```

7. Create file systems.

To create a file system, run **smitty crfs** and select Add an Enhanced Journaled File System → Add an Enhanced Journaled File System on a Previously Defined Logical Volume. The JFS2 file system `glvm_fs` is created in our example.

Set Mount AUTOMATICALLY at system restart to no (Example 15-10).

Example 15-10 Configure file systems

File system name	/glvm_fs
NEW mount point	[/glvm_fs]
SIZE of file system	
Unit Size	512bytes
+	
Number of units	[4194304]
#	[]
Mount GROUP	
Mount AUTOMATICALLY at system restart?	no
+	
PERMISSIONS	read/write
+	

```

Mount OPTIONS          []
+
Start Disk Accounting?      no
+
Block Size (bytes)        4096
Inline Log?                no
Inline Log size (MBytes)   [0]
#
Extended Attribute Format [v1]
Enable Quota Management?  no
+
Allow Small Inode Extents? yes
+
Enable EFS?               no

```

8. Create a GLVM copy.

Create a GLVM copy for each of the defined logical volumes by running **smitty glvm_utils** and selecting Geographically Mirrored Logical Volumes → Add a Remote Site Mirror Copy to a Logical Volume and set the following values:

- Select the appropriate logical volume, the remote location, and the remote physical volume (virtual disk name).
- Set NEW TOTAL number of logical partition copy to 2.
- Set Allocate each logical partition copy on a SEPARATE physical volume should be set to superstrict (Example 15-11).

Example 15-11 Creating GLVM copy

```

* LOGICAL VOLUME name           glvm_lv
* NEW TOTAL number of logical partition copies  2
+
* REMOTE PHYSICAL VOLUME name      hdisk3
  POSITION on physical volume    outer_middle
+
  RANGE of physical volumes       minimum
+
  Allocate each logical partition copy on a SEPARATE  superstrict
  physical volume?
  SYNCHRONIZE the data in the new logical partition  no
+
  copies?

```

```
* LOGICAL VOLUME name          glvm_log
* NEW TOTAL number of logical partition copies    2
+
* REMOTE PHYSICAL VOLUME name          hdisk3
  POSITION on physical volume        outer_middle
+
  RANGE of physical volumes         minimum
+
  Allocate each logical partition copy on a SEPARATE superstrict
  physical volume?
  SYNCHRONIZE the data in the new logical partition no
+
  copies?
```

9. Import GLVM Volume groups on to the remote node.

When you finish creating volume groups, logical volumes, file systems, and the GLVM copy on localnode, import the GLVM volume group imported on to remotenode. Run the following commands:

- **umount /glvm_fs file system**
- **varyoffvg glvm_vg**

On localnode, the RPV client changes from the available state to the defined state.

Then, on remotenode, the RPV server changes from the available state to the defined state

Then, on localnode, the RPV server changes from the defined state to the available state.

Finally, on remotenode, the RPV client changes from the defined state to the available state.

After you verify that all the resources are in the correct states, import the GLVM volume group by running the following command:

```
importvg -y glvm_vg -V 34 hdisk2
```

If the import is successful, varyon the volume group by running the following command:

```
varyonvg glvm_vg
```

10. Completing the environment setup.

- glvm_vg should be operational on remotenode.

- DB2 10.1 is installed and configured on both localnode and remotenode. Because the location of the DB2 file sets is standard on AIX (/opt/IBM/db2/V10.1), which is located on the local hdisk0 for both nodes, the installations can be performed in parallel. Standard installation options are chosen, and the instance is created and started automatically. DB2 is not aware of GLVM mirroring and it should not be.
- A SAMPLE database is created on the glvm_fs file system by running the following command:

```
db2sampl -dbpath /glvm_fs
```

When the SAMPLE database is created, you can perform simple tests on it, such as listing the created tables and listing the contents of the selected tables to verify that DB2 installed correctly and operates satisfactory.

15.4 Manual failover

In this section, we describe the procedure for the manual failover of the GLVM mirrored volume group.

Complete the following steps:

1. On remotenode:
 - a. Stop the DB2 instance and DB2 administration server by running the following commands:
 - i. As the instance owner, run the following commands:

```
db2stop  
db2terminate
```
 - ii. As DB2 administration server owner, run the following command:

```
db2admin stop
```
 - b. As root, run the following commands:

```
umount /glvm_fs  
varyoffvg glvm_vg
```

On remotenode, the RPV client changes from the available state to the defined state.

Then, on localnode, the RPV server changes from the available state to the defined state.

Then, on the remotenode, the RPV server changes from the defined state to the available state.

Finally, on the localnode, the RPV client changes from the defined state to the available state.

2. Run **varyonvg glvm_vg**.

3. Run **mount /glvm_fs**.

Now the GLVM volume group belongs to localnode and can be verified by running using the GLVM commands **rpvstat** and **gmvgstat** (Example 15-12).

Example 15-12 Remote Physical Volume Statistics

```
[root@NC047089.kraklab.pl.ibm.com]#/ # rpvstat
```

Remote Physical Volume Statistics:

	Comp Errors	Reads	Comp Writes	KBRead	Comp KBWrite
RPV Client	cx	Pend Reads	Pend Writes	Pend KBRead	Pend KBWrite
hdisk3	1	100	1140	801	9748
0		0	0	0	0

```
[root@NC047089.kraklab.pl.ibm.com]#/ # gmvgstat
```

GMVG Sync	Name	PVs	RPVs	Tot Vols	St Vols	Total PPs	Stale PPs
	glvm_vg	1	1	2	0	2542	0
	100%						

Because the SAMPLE database is created on the /glvm_fs file system while glvm_vg was operational on remotenode, the SAMPLE database must be catalogued on localnode as the instance owner. To accomplish this task, run the following commands:

- ▶ **db2start**
- ▶ **db2 catalog database SAMPLE on /glvm_fs**

15.5 Configuring PowerHA with GLVM

The manual process of failover is prone to error and requires specific actions to be taken at both local and remote nodes in a specific order. You should replace the manual process with scripts.

More automatic processing can be achieved by fully using PowerHA and incorporating the GLVM volume group as a resource controlled by PowerHA. GLVM volume groups can be part of the PowerHA resources under the control of PowerHA. GLVM volume groups must be defined, created, and operational before adding them to a PowerHA resource group. PowerHA recognizes them correctly and calls the appropriate events for their processing.



A

PowerHA application server scripts

In this appendix, we provide application server scripts that are used in an HADR with PowerHA SystemMirror environment. The application server start and stop scripts are used to automate the HADR takeover.

This appendix covers the following topics:

- ▶ hadr_primary_takeover.ksh
- ▶ hadr_primary_stop.ksh
- ▶ hadr_monitor.ksh

A.1 hadr_primary_takeover.ksh

Example A-1 shows a sample script for HADR takeover in an HADR with PowerHA 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 PowerHA 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/c1RGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo PowerHA 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="FAILOVER"
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"
;;
FAILOVER)
    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 PowerHA
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 neither 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 PowerHA 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/clRGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo PowerHA 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 $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 PowerHA in graceful mode"
;;

```

```

FALLOVER)
    echo "Stop PowerHA in takeover mode"

    if [[ "${ROLE_SNP_LOCAL}" = "Primary" && "${STATE_SNP_LOCAL}" = "Peer" ]]; then
        echo "HADR takeover is 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
```



B

IBM Tivoli System Automation for Multiplatforms takeover scripts

In this appendix, we provide custom takeover scripts that can be used in the HADR with IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) environment.

The start, monitor, and stop scripts are used to automate the HADR takeover.

This appendix covers the following topics:

- ▶ env file
- ▶ hadr_start.ksh
- ▶ hadr_stop.ksh
- ▶ hadr_monitor.ksh
- ▶ planned_takeover.ksh
- ▶ get_hadr_info.fnc

B.1 env file

Example B-1 shows an env definition file in an HADR with Tivoli SA MP environment.

Example: B-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}/sql1ib/db2dump/db2diag.log"
```

B.2 hadr_start.ksh

Example B-2 shows a sample script for HADR takeover in an HADR with Tivoli SA MP environment.

Example: B-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") 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") be executed."
${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME}"
elif [[ "${STATE_SNP_LOCAL}" != "Peer" ]]; then
    echo "HADR isn't in PEER state. Takeover not be executed."
    logger -i -p info -t $PROGNAME "HADR isn't in PEER state. Takeover 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
```

B.3 hadr_stop.ksh

Example B-3 shows a sample stop script for HADR takeover in an HADR with Tivoli SA MP environment.

Example: B-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

```

B.4 hadr_monitor.ksh

Example B-4 shows a sample script for monitoring the HADR state.

Example: B-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)"
        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
```

B.5 planned_takeover.ksh

Example B-5 shows a sample script for performing the takeover if a failure is detected by the monitor script.

Example: B-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.
#####

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

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 $PROGNAME "HADR ROLL is PRIMARY. This script should be executed on the standby node : Exit."
elif [[ "${ROLE_SNP_LOCAL}" = "Standby" ]]; then
    ### HADR ROLE IS STANDBY ###
    echo "### HADR_ROLE=STANDBY"
    date > ${PLANNED_TKO_FLAG}
    rgreq -o move $HADR_RG
fi

```

B.6 get_hadr_info.fnc

Example B-5 on page 552 shows a function for obtaining the HADR role and state.

Example: B-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 that are listed in this section are considered suitable for a more detailed discussion of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topic in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *IBM System Storage DS4000 and Storage Manager V10.30*, SG24-7010
- ▶ *End-to-end Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117
- ▶ *Unleashing DB2 10 for Linux, UNIX, and Windows*, SG24-8032

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources for DB2 10.1 for Linux, UNIX, and Windows:

- ▶ *Administrative API Reference*, SC27-3864-0
- ▶ *Administrative Routines and Views*, SC27-3865-00
- ▶ *Call Level Interface Guide and Reference, Volume 1*, SC27-3866-00
- ▶ *Call Level Interface Guide and Reference, Volume 2*, SC27-3867-00
- ▶ *DB2 10.1 Command Reference*, SC27-3868-00
- ▶ *DB2 10.1 Database Administration Concepts and Configuration Reference*, SC27-3871-00
- ▶ *DB2 10.1 Message Reference Volume 2*, SC27-3880-00

- ▶ *Database Administration Concepts and Configuration Reference*, SC27-3871-00
- ▶ *Data Movement Utilities Guide and Reference*, SC27-3869-00
- ▶ *Data Recovery and High Availability Guide and Reference*, SC27-3870-00
- ▶ *Database Security Guide*, SC27-3872-00
- ▶ *DB2 Workload Manager Guide and Reference*, SC27-3891-00
- ▶ *Developing ADO.NET and OLE DB Applications*, SC27-3873-00
- ▶ *Developing Embedded SQL Applications*, SC27-3874-00
- ▶ *Developing Java Applications*, SC27-3875-00
- ▶ *Developing Perl, PHP, Python, and Ruby on Rails Applications*, SC27-3876-00
- ▶ *Developing User-Defined Routines (SQL and External)*, SC27-3877-00
- ▶ *Getting Started with Database Application Development*, GI13-2046-00
- ▶ *Getting Started with DB2 Installation and Administration on Linux and Windows*, GI13-2047-00
- ▶ *Globalization Guide*, SC27-3878-00
- ▶ *Message Reference, Volume 1*, SC27-3879-00
- ▶ *Message Reference, Volume 2*, SC27-3880-00
- ▶ *Net Search Extender Administration and User's Guide*, SC27-3895-00
- ▶ *Partitioning and Clustering Guide*, SC27-3882-00
- ▶ *PowerHA SystemMirror: Master Glossary*, SC23-6757
- ▶ *pureXML Guide*, SC27-3892-00
- ▶ *QL Procedural Languages: Application Enablement and Support*
SC27-3896-00
- ▶ *RSCT for Multiplatforms: Technical Reference*, SA22-7893-19
- ▶ *Spatial Extender User's Guide and Reference*, SC27-3894-00
- ▶ *SQL Reference, Volume 1*, SC27-3885-00
- ▶ *SQL Reference, Volume 2*, SC27-3886-00
- ▶ *Text Search Guide*, SC27-3888-00
- ▶ *Tivoli System Automation for Multiplatforms Version 3.2.2 Administrator's and User's Guide*, SC34-2583-00
- ▶ *Tivoli System Automation for Multiplatforms Version 3.2.2 Installation and Configuration Guide*, SC34-2584-03

- ▶ *Troubleshooting and Tuning Database Performance*, SC27-3889-00
- ▶ *What's New*, SC27-3890-00
- ▶ *XQuery Reference*, SC27-3893-00

Online resources

These websites are also relevant as further information sources for DB2 10.1 for Linux, UNIX, and Windows:

- ▶ Database and Information Management home page:
<http://www.ibm.com/software/data/>
- ▶ DB2 developerWorks
<http://www.ibm.com/developerworks/db2/>
- ▶ DB2 Information Center:
<http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/>
- ▶ DB2 for Linux
<http://www.ibm.com/software/data/db2/linux/>
- ▶ DB2 for Linux, UNIX, and Windows Application Development
<http://www.ibm.com/software/data/db2/ad/>
- ▶ DB2 Product Family Library:
<http://www.ibm.com/software/data/db2/library/>
- ▶ DB2 Technical Support:
<http://www.ibm.com/software/data/db2/support/db29/>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM



Redbooks

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

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



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



Learn DB2 HADR setup, administration, monitoring, and preferred practices

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 using enterprise system resources with broad support for clustering software, such as IBM PowerHA SystemMirror, IBM Tivoli System Automation for Multiplatforms (SA MP), and Microsoft Windows Cluster Server.

Use PowerHA, MSWFC, Tivoli SA MP with DB2, and DB2 HADR

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

Protect data with DB2 disaster recovery options

This book explains how to configure Cluster software PowerHA, Tivoli SA MP, 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, and the db2recovery command, in detail.

This book is intended for database administrators and information management professionals who want to design, implement, and support a highly available DB2 system.

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