

Build a highly available application platform for J2EE, Part 5: Set up IBM DB2 UDB for high availability

Skill Level: Intermediate

[Joyce Coleman](#)
Software Engineer
EMC

[Steve Raspudic](#)
Software engineer
EMC

21 Jun 2005

In this tutorial, you set up IBM DB2® Universal Database™ (UDB) for high availability using High Availability Disaster Recovery (HADR), Automatic Client Reroute, and IBM Tivoli® System Automation for Multiplatforms.

Section 1. Before you start

About this tutorial

This [series](#) from the IBM Continuous Computing team focuses on building a highly available solution platform for Java™ 2 Platform, Enterprise Edition (J2EE). Follow along as the team uses existing hardware and software from across IBM divisions to produce a complete solution that offers high availability. In the first phase of this series, we use existing products to establish the baseline of what is possible using current technologies. Next, we enhance the design of the system to take advantage of emerging technologies in the areas of automation, faster failure detection, and multisite failover.

This tutorial describes how to set up IBM DB2® Universal Database™ (UDB) for high availability using High Availability Disaster Recovery (HADR) software, Automatic Client Reroute, and IBM Tivoli® System Automation for Multiplatforms.

If you want to learn how to set up DB2 for high availability, take this tutorial. Basic knowledge of DB2 UDB and the Linux™ operating system will help you complete the tasks.

Prerequisites

To set up the environment described in this tutorial, you'll need:

- A three-node cluster on IBM eServer™ xSeries® hardware
- Redundant network adapters (NICs)
- Sufficient logical disks (LUNs) to host all database data
- IBM Tivoli System Automation (TSA) for Multiplatforms Version 1.2
- DB2 UDB Enterprise Server Edition Version 8.2 or later
- Red Hat Enterprise Linux 3.0
- The first node hosts our primary database instance, the second node hosts our standby database instance, and the third node is used by TSA to provide quorum functionality. The third node does not need to be dedicated to this function and can be used to perform other functions or to support other applications. If a shared disk is available, it can be used to provide quorum functionality instead of the third node. (See the Resources section for information about this configuration.)

Section 2. Introduction

About this project

The charter of the IBM Continuous Computing team is to build a highly available solution platform for J2EE applications that provides end-to-end availability. In the first phase, you learn how to integrate and configure existing IBM hardware and software products to understand what is possible using current technologies. Next, we enhance the design of the system to take advantage of emerging technologies in automation, faster failure detection, and multisite failover.

This tutorial, the fifth installment in our [series](#), describes how to set up DB2 UDB for high availability (HA) using High Availability Disaster Recovery (HADR) software, Automatic Client Reroute, and IBM Tivoli System Automation for Multiplatforms (IBM Tivoli SAM).

Review the HA features of DB2 UDB

What is a highly available database management system? The software that powers the database must always be running and available for transaction processing, even when there is a hardware or software failure, or when disaster strikes and the entire server site is destroyed.

DB2 UDB offers many features that can support a highly available solution. Several features make it possible to maintain a copy of a database on a separate server that is continuously rolling log files forward to maintain a duplicate of the database. If the primary database fails, any remaining log files are transferred to the standby machine. The standby database rolls forward to the end of the logs and stops, and clients reconnect to the standby machine.

Several highly available cluster managers can be used with DB2 UDB to transfer the workload of a failed machine to a backup machine. As we saw in [Part 3](#) of this series, you can set up a primary and standby machine, each of which has access to the shared storage on which the database is located. If the primary machine fails, its IP address and DB2 instance can be failed over to the backup machine, allowing clients to continue accessing the data with minimal disruption.

An additional feature that can provide high availability in a UNIX® environment is the fault monitor. When the fault monitor is used, any instance that exits prematurely is restarted automatically.

Set up HADR, Automatic Client Reroute, and IBM Tivoli SAM

High Availability Disaster Recovery (HADR) is a data replication feature introduced with DB2 UDB V8.2 that provides a highly available solution for both partial and complete site failures. HADR protects against data loss by replicating data changes from a source database, called the primary, to a target database, called the standby. It is an example of database log-shipping technology, where transactions that occur on the primary database are applied to a designated standby database. If the primary database becomes inaccessible, the standby database can take over the role of primary and clients can switch over to using this database.

To allow client applications to switch over automatically to the standby database if the primary database becomes inaccessible, we will use a second DB2 feature called Automatic Client Reroute. Under normal circumstances, a client application

that loses communication with a database server receives an error message and stops trying to connect to the database. When Automatic Client Reroute is used, the client application is instead rerouted to an alternate server.

For the standby database to take over as the new primary and begin servicing requests from clients, it is necessary to issue a `takeover` command. Until this command is issued, the standby database remains in standby mode, even if the primary instance fails, and clients cannot connect to it. One final piece is therefore needed to complete this highly available solution, and that is the automation of the `takeover` command. We will use IBM Tivoli System Automation for Multiplatforms (IBM Tivoli SAM) to monitor the cluster members and to run a takeover script if it detects that the node hosting the primary instance has failed.

Section 3. Tivoli SAM, HADR, and Automatic Client Reroute

Set up Tivoli SAM to manage DB2 resources

In [Part 3](#), we set up IBM Tivoli SAM to manage DB2 resources for a single-instance failover. We'll now repeat the steps, but this time so that IBM Tivoli SAM can manage our two HADR databases.

1. Prepare all nodes that will comprise the cluster. Issue the following command on the nodes, which we assume here are named `halin1`, `halin2` and `halin3`:

```
# /usr/sbin/rsct/bin/preprnode halin1 halin2 halin3
```

2. Create a cluster domain called `hadr_domain` that includes the three nodes:

```
# /usr/sbin/rsct/bin/mkrpdomain hadr_domain halin1 halin2 halin3
```

3. Make the domain active:

```
# /usr/sbin/rsct/bin/starttrpdomain hadr_domain
```

To ensure that the domain is online, issue the command:

```
# /usr/sbin/rsct/bin/lsrpdomain
```

You should see information similar to the following:

Name	OpState	RSCTActiveVersion	MixedVersions	TSPort
hadr_domain	Online	2.4.1.2	No	12347

To ensure all nodes in the domain are online, issue the command:

```
# lsrpnode
```

Output similar to the following should be displayed:

Name	OpState	RSCTVersion
halin1	Online	2.4.1.2
halin2	Online	2.4.1.2
halin3	Online	2.4.1.2

Create the DB2 instances

- As outlined in the DB2 installation guide, create the necessary user groups and IDs. Note that we named our instance owners db2instp (on halin1) and db2insts (on halin2). Use the `db2icrt` command to create an instance on each machine. We named our instances db2instp (on halin1) and db2insts (on halin2). Assuming that the instances are successfully created, you will see the following output:

```
DBI1070I Program db2icrt completed successfully.
```

- Make sure the `/etc/services` file is set up correctly and that the relevant entries are identical across both nodes by issuing the following commands:

```
# rsh halin1 "cat /etc/services | grep inst"
```

```
# rsh halin2 "cat /etc/services | grep inst"
```

The output should look like the following:

xinstp	18817/tcp
--------	-----------

xinsts	18818/tcp
hadrintp	18819/tcp
hadrints	18820/tcp

The name used for the services entry is not critical, but it ensures that four entries with port numbers unique from all other entries in the `/etc/services` file exist.

3. Ensure that you have unprompted rsh access (as root) to the other nodes in this cluster. Unprompted rsh access to the other nodes is only required for the duration of the registration process and can be revoked once the registration script has completed processing.
4. Create an equivalency group for the network interface cards (NICs) at each node. To determine the names of the NICs to include in the equivalency group, first use the `ifconfig` command to list the available NICs on each node.
For example, at `halin1` issue the following command:

```
# ifconfig -a | egrep 'eth|Mask:'
```

The following is displayed in response on our system:

```
eth0 Link encap:Ethernet
inet addr:8.26.96.150 Bcast:8.26.97.255

eth1 Link encap:Ethernet
inet addr:192.168.1.1 Bcast:192.168.1.255

eth2 Link encap:Ethernet
inet addr:192.168.1.2 Bcast:192.168.1.255

eth3 Link encap:Ethernet
inet addr:8.26.96.56 Bcast:8.26.97.255
```

The second node, `halin2`, contains the same set of adapters though the output is not shown here. Thus, any of the set of adapters `eth0`, `eth1`, `eth2` or `eth3` are available at `halin1` to host the public HA IP address. Choose at least two of these, with the condition being that all adapters chosen must share the identical netmask and all must be accessible from the clients. In this cluster, we will consider the adapters `eth0` and `eth3` on `halin1`, and adapters `eth0` and `eth3` on `halin2` as equivalent for the purposes of hosting the IP addresses.

To express this equivalency, issue the following commands on `halin1`:

```
# mkequ virpubnic_halin1 IBM.NetworkInterface:eth0:halin1,eth3:halin1
```

```
# mkequ virpubnic_halin2 IBM.NetworkInterface:eth0:halin2,eth3:halin2
```

Then issue the following commands, again on halin1:

```
# cd /opt/IBM/db2/V8.1/ha/salinux
```

```
# ./regdb2salin -a db2instp -r -i 8.26.96.84
```

The `-r` option ensures that the instance will restart in place and not failover to any other node in the cluster. The `-i` option specifies the highly available IP address to associate with this instance. Any unused IP address can be chosen, but it must be on the same netmask as the IP addresses of the base adapters.

5. Set up a dependency between the HA IP resource and the network equivalency group. Make the resource group go offline as follows:

```
# chrg -o Offline db2_db2instp_0-rg
```

Then, create the dependency relationship:

```
# mkrel -p DependsOn -S IBM.ServiceIP:db2_db2instp_0-rs_ip
-G IBM.Equivalency:virpubnic_halin1 db2_db2instp_0-rs_ip_equiv_do
```

6. Do the same steps for the DB2 instance on halin2:

```
# cd opt/IBM/db2/V8.1/ha/salinux
```

```
# ./regdb2salin -a db2insts -r -i 8.26.97.10
```

```
# chrg -o Offline db2_db2insts_0-rg
```

```
# mkrel -p DependsOn -S IBM.ServiceIP:db2_db2insts_0-rs_ip
-G IBM.Equivalency:virpubnic_halin2 db2_db2insts_0-rs_ip_equiv_do
```

7. Bring the resource groups online:

```
# chrg -o Online db2_db2instp_0-rg
```

```
# chrg -o Online db2_db2insts_0-rg
```

Create the Trade6 database

We will now create the Trade6 database. It is not possible to populate the database before WebSphere Application Server is installed and the Trade6 application has been deployed, so we will instead set up HADR for the unpopulated database.

1. Download and unzip the Trade6 zip file at <http://pulsar.raleigh.ibm.com/downloads.html>. This creates a new directory on halin1 called tradeinstall. From the tradeinstall directory, as the DB2 user (db2instp), execute the following commands to create the database:

```
# db2start

# mkdir /misc/homep/trade6

# db2 create db tradedb on /misc/homep/trade6

# db2 connect to tradedb

# db2 -tvf DB2/Table.ddl
```

You can check for the successful creation of the database by listing the database directory:

```
# db2 list db directory
```

If the command is successful, you should see information similar to the following:

Database 1 entry:	
Database alias	= TRADEDB
Database name	= TRADEDB
Local database directory	= /misc/homep/trade6
Database release level	= a.00
Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	=

```
Alternate server port number
```

```
=
```

2. Turn on LOGRETAIN for this database and take a database backup image as follows:

```
# db2 update db cfg for tradedb using LOGRETAIN ON
```

```
# db2 backup db tradedb to /misc/homep/backup
```

```
# db2 disconnect all
```

```
# db2stop
```

```
# db2start
```

3. FTP the database backup to halin2.

Set up the standby database

1. On halin2, as the DB2 user (db2insts), restore the backup copy of tradedb to halin2 by executing the following commands:

```
# db2start
```

```
# mkdir /misc/homes/trade6
```

```
# db2 restore db tradedb from <backup_db_location> to  
/misc/homes/trade6 replace history file
```

Keep in mind that you should absolutely *not* perform a redirected restore. The database must be created using the same files, devices, or file systems if using absolute paths.

2. Issue the following commands as user db2instp on node halin1:

```
# db2set DB2COMM=tcPIP
```

```
# db2 update dbm cfg using SVCENAME xinstp
```

Issue the following commands as user db2insts on node halin2:

```
# db2set DB2COMM=tcpip
```

```
# db2 update dbm cfg using SVCENAME xinsts
```

3. Update the database-level configuration parameters required for an HADR pair to be established. For example, as user db2instp on halin1, issue the following commands:

```
# db2 update db cfg for tradedb using HADR_LOCAL_HOST 8.26.96.84
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_HOST 8.26.97.10
```

```
# db2 update db cfg for tradedb using HADR_LOCAL_SVC 18819
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_SVC 18820
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_INST db2insts
```

Note that the values for the HADR_LOCAL_SVC and HADR_REMOTE_SVC are taken from the values we found in the /etc/services file earlier.

4. Update the parameters as user db2insts on halin2, as follows:

```
# db2 update db cfg for tradedb using HADR_LOCAL_HOST 8.26.97.10
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_HOST 8.26.96.84
```

```
# db2 update db cfg for tradedb using HADR_LOCAL_SVC 18820
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_SVC 18819
```

```
# db2 update db cfg for tradedb using HADR_REMOTE_INST db2instp
```

HADR_LOCAL_HOST and HADR_LOCAL_SVC on one database must match HADR_REMOTE_HOST and HADR_REMOTE_SVC on the other base, and vice versa.

5. Update the alternate server at each instance. This is required in order to support the Automatic Client Reroute feature. For instance db2instp, enter the following command on halin1:

```
# db2 update alternate server for database tradedb using
                               hostname 8.26.97.10 port 18818
```

For instance db2insts, issue the following command on halin2:

```
# db2 update alternate server for database tradedb
                               using hostname 8.26.96.84 port 18817
```

6. As instance db2insts, start the tradedb on halin2 as a standby database:

```
# db2 start hadr on db tradedb as standby
```

Then, as instance db2instp, start the tradedb on halin1 as an HADR primary:

```
# db2 start hadr on db tradedb as primary
```

7. Once this is complete, ensure that the HADR pair is in peer state with the instance db2instp hosting the primary database tradedb on the physical host halin1, and the instance db2insts hosting the standby database tradedb on the physical host node2. You can verify this using the DB2 `snapshot` command from the primary node:

```
# db2 get snapshot for all on tradedb | more
```

Partway through the output you should see information similar to the following:

HADR Status	
Role	= Primary
State	= Peer
Synchronization mode	= Sync
Connection status	= Connected
Remote instance	= db2insts
Timeout (seconds)	= 30
Primary log position (file, page, LSN)	= S0000000.LOG, 0, 00000000007D0000
Standby log position (file, page, LSN)	= S0000000.LOG, 0, 00000000007D0000

Create the HADR resource group

Now we must configure the resource group controlling the HADR state. We'll register the resource group at the node currently hosting the primary, which in our example is `halin1`.

1. Issue the following command sequence to register the HADR pair:

```
# rlogin halin1
```

```
# cd /opt/IBM/db2/V8.1/ha/salinux
```

```
# ./reghadrsalin -a db2instp -b db2insts -d tradedb
```

The argument to `-a` is the name of the instance hosting the HADR primary database, the argument to `-b` is the name of the instance hosting the HADR standby database, and the argument to `-d` is the name of the HADR database itself.

The HADR pair state is now controlled by the cluster manager. To view the cluster status, issue the `getstatus` command:

```
# getstatus
```

You should see output similar to the following:

Resource Groups and Resources

Group Name	Resources
db2_db2instp_0-rg	db2_db2instp_0-rs
db2_db2instp_0-rg	db2_db2instp_0-rs_ip
db2_db2insts_0-rg	db2_db2insts_0-rs
db2_db2insts_0-rg	db2_db2insts_0-rs_ip
db2hadr_tradedb-rg	db2hadr_tradedb-rs

Resource Name	Node Name	State
db2_db2instp_0-rs	halin1	Online
db2_db2instp_0-rs_ip	halin1	Online
db2_db2insts_0-rs	halin2	Online
db2_db2insts_0-rs_ip	halin2	Online

db2hadr_tradedb-rs	halin1	Online
db2hadr_tradedb-rs	halin2	Offline

If the physical host halin1 now fails, the resource group db2hadr_tradedb-rg will failover to be hosted by physical host halin2, which will cause the HADR primary database to be hosted on halin2.

Do a controlled takeover

1. Change the location of the node hosting the HADR primary database with the following command:

```
# rgreq -o Move -n halin1 db2hadr_tradedb-rg
```

Issue a `getstatus` command and see that the HADR primary database is now hosted by the machine halin2.

2. To bring the HADR primary database back to being hosted by halin1, verify that the HADR pair is in peer state and issue the command:

```
# rgreq -o Move -n halin2 db2hadr_tradedb-rg
```

Once `getstatus` indicates that the HADR primary database is hosted by instance db2instp, issue a `DB2 snapshot` command. Verify that the instance db2instp is hosting the HADR primary database tradedb and that db2insts is hosting the HADR standby database tradedb. Also verify that the databases are in peer state.

You can only use the command `rgreq -o Move -n node_name db2hadr_tradedb-rg` to do the controlled failover. You should not use the `takeover` command directly when the cluster manager is in control of the HADR pair.

Section 4. Sample failure scenarios

Instance failure

For all of the tests in this section, it is recommended that client load be active against the database system, similar to what could be expected in the production environment. Because that is not possible right now, as we have not yet installed WebSphere Application Server or deployed the Trade application, you might want to return to these sample failure scenarios after you are able to run the Trade workload.

To simulate an instance failure on the primary node, log in to `halin1` as the instance owner `db2instp` and issue the following command:

```
# kill -9 $(ps -ef | grep db2sysc | grep -v grep | awk '{print $2}')
```

Issue a `getstatus` command repeatedly, and observe that the resource for the DB2 instance goes offline briefly in response to the command and is quickly restarted on the same node. Verify that the HADR pair has reached peer state.

To simulate an instance failure on the standby node, log in to `halin2` as the instance owner `db2insts` and issue the preceding command.

Issue a `getstatus` command repeatedly, and observe that the resource for the DB2 instance goes offline briefly in response to the command and is quickly restarted on the same node. Note also that the HADR resource group state goes offline briefly and is restarted within the monitor period of the HADR resource group.

Machine failure

Turn off the power on the primary machine (or alternately, issue as root the `reboot -f` command). The resource group `db2hadr_tradedb-rg` will failover to node `halin2`, causing instance `db2insts` to host the HADR primary for database `tradedb`.

For this type of failover, where the old primary is completely inaccessible, the resource group start script may have issued a `takeover by force` command. If this is the case, the resource group controlling instance `db2instp` (in this case, named `db2_db2instp_0-rg`) is brought offline to prevent any possible "split-brain" scenario from occurring. This ensures that the instance `db2instp` does not come online without some operator intervention. Follow the steps in the next section to reestablish the HADR pair.

If the standby machine is powered off, the clients pause for a time up to the `HADR_TIMEOUT` configuration parameter. After this length of time elapses, the HADR state becomes Primary Disconnected on the primary machine, and the clients continue processing. Once the standby machine is brought back online, the standby instance is also brought online. Verify that HADR peer state is attained.

Public adapter failure

If a single public network adapter on either the primary or standby machine fails, you should see at most a brief interruption of service while the IP address fails over to the other adapter or adapters in that equivalency group. You can simulate this by pulling the cable on one of the public adapters.

If all the adapters in the equivalency group on the primary node fail, IBM Tivoli SAM fails over the HADR resource group to the standby node and the state of the failed node is `failed_offline`. To re-integrate the former primary as the current standby, you will have to follow the re-integration steps documented in the next section.

If all the adapters in the equivalency group on the standby node fail, the node is shut down. As in the case where the standby machine is powered off, the HADR state on the primary node becomes `Primary Disconnected` and clients continue processing. Once the standby machine is brought back online, the standby instance is also brought online.

Section 5. Additional considerations

Start and stop DB2 instances

To stop and start the db2 instances, you must follow this procedure:

```
# chrg -o Offline db2hadr_tradedb-rg
```

```
# chrg -o Offline db2_db2instp_0-rg
```

```
# chrg -o Offline db2_db2insts_0-rg
```

Then bring the instances online again using the following commands:

```
# chrg -o Online db2insts_0-rg
```

```
# chrg -o Online db2instp_0-rg
```

Finally, start up the HADR pair and ensure that the nodes reach peer state. Once that is achieved, you may issue the following command to have IBM Tivoli SAM

control the state of the resource group:

```
# chrg -o Online db2hadr_tradedb-rg
```

As mentioned earlier, you can only use the command `rgreq -o Move -n node_name db2hadr_tradedb-rg` to do the controlled failover. You should not use the `takeover` command directly.

Reestablish HADR

In some failover scenarios, when the old primary is completely inaccessible, the resource group start script might issue a `takeover by force` command. If this is the case, the resource group controlling instance `db2instp` is brought offline. Special steps are required to reestablish the HADR pair once the primary node has come back online.

As instance owner `db2instp` on `halin1`, disable TCP/IP communications so that clients are unable to access the database.

```
# db2set DB2COMM=
```

Then, as the root user, bring the resource group online:

```
# chrg -o Online db2_db2instp_0-rg
```

As instance owner `db2instp`, start up the database in the role of standby:

```
# db2 start hadr on db tradedb as standby
```

Verify that HADR peer state has been achieved. Then, enable TCP/IP:

```
# db2set DB2COMM=tcPIP
```

The HADR pair should now be reestablished. Verify that it has reached peer state.

To bring the primary back to being hosted by the instance `db2instp`, move the HADR resource group to node `halin1` using the following command:

```
# rgreq -o Move -n halin2 db2hadr_tradedb-rg
```

Successful re-integration is not guaranteed after a `takeover by force` command if at the time of the failure the HADR pair was not in peer state. If the re-integration steps given above fail to establish an HADR pair in peer state, then it is necessary to create a new standby database using an image backup of the current primary

database.

Section 6. Summary and resources

Summary

This tutorial demonstrated the steps for setting up a highly available database:

1. Set up IBM Tivoli SAM to manage the DB2 resources.
 - Create the cluster domain and bring it online.
 - Create the tiebreaker object.
3. Create the DB2 instances and the database.
 - Create one instance on the primary node and another on the standby node.
 - Create an equivalency group for the network interfaces on each node.
 - Set up a dependency between the IP resources and the network equivalency groups.
 - Create the Trade6 database and back it up.
5. Set up the standby database.
 - Restore the database on to the standby machine.
 - Update the necessary registry variables, database configuration parameters, and database manager configuration parameters, as well as the alternate server information.
 - Start HADR on the standby and then primary machines.
7. Create the HADR resource group.
 - Register the resource group at the primary node.
9. Do a controlled failover to test the system.

Resources

Learn

- For a more detailed discussion on how to set up DB2 UDB and IBM Tivoli System Automation for Multiplatforms for high availability, read [Automating DB2 HADR Failover on Linux using Tivoli System Automation for Multiplatforms](#).
- To learn more about DB2's high-availability features, use the [Data Recovery and High Availability Guide and Reference](#).
- [IBM DB2 Universal Database Administration Guide](#) contains general information about DB2 UDB.
- For more information about installing IBM Tivoli System Automation for Multiplatforms, use the [IBM Tivoli System Automation for Multiplatforms Guide and Reference Version 1.2](#).

Get products and technologies

- Download and the Trade6 zip file at <http://pulsar.raleigh.ibm.com/downloads.html>.

About the authors

Joyce Coleman

Joyce Coleman is a member of the IBM DB2/WebSphere performance team. She is currently working on the IBM autonomic computing benchmark. Contact Joyce at colemanj@ca.ibm.com.

Steve Raspudic

Steve Raspudic is a DB2 Information Management Software engineer for the IBM Software Group. For technical questions or comments about the content of this tutorial, contact Steve at stevera@ca.ibm.com.