VERITAS™

VERITAS Cluster Server Enterprise Agent 4.0 for DB2

Installation and Configuration Guide

AIX

N10188J

August 2004

Disclaimer

VERITAS Legal Notice

VERITAS Software Corporation
350 Ellis Street
Mountain View, CA 94043
USA
Phone 650–527–8000 Fax 650–527–2908
http://www.veritas.com

Contents

Preface

This document provides instructions on how to install and configure the VERITAS Cluster Server Agent for DB2 UDB.

How This Guide Is Organized

Chapter 1, "Introduction" on page 1 introduces you to the VERITAS Cluster Server (VCS) agent, version 4.0, for DB2 UDB.

Chapter 2, "Preparation for Installing DB2 UDB in a VCS Cluster" on page 9 describes how to prepare for the installation.

Chapter 3, "Installing the DB2 UDB Agent Software" on page 19 describes installing the agent.

Chapter 4, "Configuring the DB2 Agent" on page 21 describes the configuration steps for the agent.

Chapter 5, "Disabling and Removing the Agent" on page 39 describes how to disable and remove the agent.

Appendix A, "Sample Configuration Files" on page 41 gives you sample configuration files for the agent.

Conventions

Convention	Usage	Example
`monospace`	Used for path names, commands, output, directory and file names, functions, and parameters.	Read tunables from the `/etc/vx/tunefstab` file. See the `ls`(1) manual page for more information.
`monospace` (**`bold`**)	Indicates user input.	# **`ls pubs`** C:\> **`dir pubs`**
italic	Identifies book titles, new terms, emphasized text, and variables replaced with a name or value.	See the *User's Guide* for details. The variable *system_name* indicates the system on which to enter the command.
bold	Depicts GUI objects, such as fields, list boxes, menu selections, etc. Also depicts GUI commands.	Enter your password in the **Password** field. Press **Return.**
blue text	Indicates hypertext links.	See "Getting Help" on page ix.
#	Unix superuser prompt (all shells).	# **`cp /pubs/4.0/user_book /release_mgnt/4.0/archive`**

Getting Help

For technical assistance, visit http://support.veritas.com and select phone or email support. This site also provides access to resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service. Use the Knowledge Base Search feature to access additional product information, including current and past releases of VERITAS documentation.

Additional Resources

For license information, software updates and sales contacts, visit https://my.veritas.com/productcenter/ContactVeritas.jsp. For information on purchasing product documentation, visit http://webstore.veritas.com.

Documentation Feedback

Your feedback on product documentation is important to us. Send suggestions for improvements and reports on errors or omissions to clusteringdocs@veritas.com. Include the title and part number of the document (located in the lower left corner of the title page), and chapter and section titles of the text on which you are reporting. Our goal is to ensure customer satisfaction by providing effective, quality documentation. For assistance with topics other than documentation, visit http://support.veritas.com.

Introduction

1

Welcome to the VERITAS Cluster Server (VCS) agent, version 4.0, for DB2 UDB. DB2 Universal Database is a relational database management system. This guide describes the agent for DB2 UDB, its modes of operation, and its attributes. It describes how to install and configure the agent.

Supported Software

The VCS enterprise agent, version 4.0 for DB2 UDB, supports DB2 Universal Database Enterprise Server Edition (ESE) versions for single and multi-partition instance. The agent supports ESE for single partition instance for versions 7.2 and 8.1. For ESE multi-partition instance, it supports the Symmetric Multiprocessing (SMP) hardware configuration for versions 7.2 and 8.1 and supports the Massively Parallel Processing (MPP) hardware configuration for version 8.1. The agent supports AIX version 5L v.5.2. The VCS version must be 4.0 and above. The memory requirements vary for different versions of DB2 being used. Check the *DB2 Universal Database Quick Beginnings Guide* for information about memory requirements.

About the DB2 UDB Agent

The DB2 UDB enterprise agent monitors DB2 database instances while they are up and running on a given system. If the system fails, the agent detects the failure and takes the DB2 instances offline. VCS conducts failover to another system in the cluster, where the agent brings DB2 instances online.

For ESE single partition instance, the agent brings DB2 UDB database instances online, monitors database processes, and shuts down instance(s). For ESE multi-partition instance, it brings DB2 UDB database partitions online, monitors the database processes at the partition level, and shuts down database partitions.

Operations of the DB2 UDB Agent

The online, offline, monitor, and clean operations performed by the DB2 agent vary depending on whether the version of DB2 UDB is ESE single-partition instance or ESE multi-partition instance and the ESE version. These operations are described in the following sections.

Online Operation

For ESE versions earlier than version 8, the agent uses the `db2start` program to start a DB2 instance or database partition.

- The command for ESE single-partition instance is:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/adm/db2start"
  ```

- The command for ESE multi-partition instance in SMP hardware configuration is:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/adm/db2start $nodenum
    $NodeNumber"
  ```

For ESE version 8, the agent uses `db2gcf` program to start either the ESE single-partition instance configuration or the ESE multi-partition instance in SMP or MPP hardware configuration. The command is:

```
su $DB2InstOwner -c "$InstHome/sqllib/bin/db2gcf -u -i
    $DB2InstOwner -p $nodenum
```

Offline Operation

For ESE versions earlier than version 8, the agent uses the `db2stop` program to stop a DB2 instance or database partition.

- The command for ESE single-partition instance is:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/adm/db2stop force"
  ```

- The command for ESE multi-partition instance in SMP hardware configuration is:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/adm/db2stop force $nodenum
     $NodeNumber"
  ```

For ESE version 8, the agent uses the `db2gcf` program to stop either the ESE single-partition instance configuration or the ESE multi-partition instance in SMP or MPP hardware configuration. The command is:

```
su $DB2InstOwner -c "$InstHome/sqllib/bin/db2gcf -d -i
      $DB2InstOwner -p $nodenum
```

Monitor Operation

The commands used by the agent to monitor the DB2 instances vary depending on the DB2 version and hardware configuration.

ESE Single-Partition Instance (Earlier than Version 8)

For versions of the ESE single-partition instance earlier than version 8, the agent executes the command:

```
db2nps 0
```

to check the processes owned by the instance owner. If the output of the command contains DB2 processes, the monitor exits with a return value of 110, indicating success. If this command doesn't exist, then the monitor tries to find the `db2sysc` process owned by the instance in the process table. If the monitor finds the process in the table, it exits with a return value of 110 and the DB2 instance continues online. Otherwise, the DB2 instance is taken offline and failed over to the next system in the service group's `SystemList` attribute (if the `RestartLimit` and `ToleranceLimit` are set to 0).

If in-depth monitoring is enabled (that is, the `IndepthMonitor` attribute is set to 1 and the `DatabaseName` attribute is not NULL), the monitor performs a query to the database indicated in the `DatabaseName` attribute. If the query succeeds, the instance continues online (monitor exit code for success is 110).

If the database query has any errors or problems, it checks the value of the `WarnOnlyIfDBQueryFailed` attribute of the Db2udb agent. If `WarnOnlyIfDBQueryFailed` is set to 1 (the default), the agent logs an error message containing the actual SQL error in the engine log (no more than once a day to prevent overflowing the engine log) and returns exit code 110. If `WarnOnlyIfDBQueryFailed` is set to 0, the agent logs an error message and returns exit code 100, in which case the instance restarts or fails over.

ESE Multi-Partition Instance, SMP Hardware Configuration (Earlier than Version 8)

For versions of the ESE multi-partition instance earlier than version 8, the agent executes the command:

```
db2nps $NodeNumber
```

to check the processes owned by the instance owner for a particular database partition or node number. If the output of the command contains DB2 processes, the monitor returns an exit code of 110. Otherwise, the database partition owned by the instance is taken offline and failed over or restarted, depending on the values of the RestartLimit and ToleranceLimit attributes. If both attributes are set to 0, the instance fails over to the next system in the group's `SystemList` attribute.

If in-depth monitoring is enabled (that is, the `IndepthMonitor` attribute is set to 1 and the `DatabaseName` attribute is not NULL), the monitor performs a query to the database indicated in the `DatabaseName` attribute. The database is to be created from this partition. The database partition must be able to connect to this local database it monitors without error. The default MonitorTimeout value is 240 seconds. If the database connection you use is very slow, you can adjust the MonitorTimeout attribute to a higher value. If the query succeeds, `monitor` returns exit code 110 (`monitor` returns exit 110 for online and 100 for offline).

If the database query has any errors or problems, it checks the value of theDb2udb agent's `WarnOnlyIfDBQueryFailed` attribute. If `WarnOnlyIfDBQueryFailed` is set to 1 (the default), the agent logs an error message containing the actual SQL error in the engine log (no more than once a day to prevent overflowing the engine log) and returns exit code 110. If `WarnOnlyIfDBQueryFailed` is set to 0, the agent logs an error message and returns exit code 100.

ESE Configurations, Version 8.0 or Later

For all DB2 ESE configurations, version 8.0 or later, for a single or multi-partition instance in SMP or MPP configuration, the agent executes the command:

```
db2gcf -s -i $DB2InstOwner -p $nodenum
```

to check the status of the database partition or node number. If the exit status of the `db2gcf` command is 0, `the` monitor returns exit code 110. Otherwise, the monitor returns an exit code of 100 and the resource is taken `offline`.The agent then restarts or fails over the resource, depending on other type-independent attributes, such as `RestartLimit` or `ToleranceLimit`. The command `db2gcf` is only available.

If in-depth monitoring is enabled (that is, the `IndepthMonitor` attribute is set to 1 and the `DatabaseName` attribute is not NULL, the monitor performs a query to the database indicated in the `DatabaseName` attribute. The database is to be created from this partition.

If the database query has any errors or problems, it checks the value of the Db2udb agent's `WarnOnlyIfDBQueryFailed` attribute. If `WarnOnlyIfDBQueryFailed` is set to 1 (the default), the agent logs an error message containing the actual SQL error in the engine log (no more than once a day to prevent overflowing the engine log) and returns exit code 110. If `WarnOnlyIfDBQueryFailed` is set to 0, the agent logs an error message and returns exit code 100. The default MonitorTimeout value is 240 seconds. If the database connection you use is very slow, you can adjust the MonitorTimeout attribute to a higher value.

Clean Operation

For ESE versions earlier than version 8, the agent uses the `db2nkill` program with instance owner's ID to kill the instance processes owned by the $DB2InstOwner for the instance or database partition.

- For ESE single-partition instance, the agent uses the following command to kill a DB2 instance:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/bin/db2nkill 0"
  ```

- For ESE multi-partition instance, the agent uses the following command to kill a DB2 database partition in the SMP configuration:

  ```
  su $DB2InstOwner -c "$InstHome/sqllib/bin/db2nkill nodenum
     $NodeNumber"
  ```

For ESE version 8, the agent uses the `db2gcf` program to kill a DB2 database partition with either the ESE single-partition instance configuration or the ESE multi-partition instance in SMP or MPP configuration. The command is:

```
su $DB2InstOwner -c "$InstHome/sqllib/bin/db2gcf -k -i
   $DB2InstOwner -p $nodenum
```

Info Entry Point Operation

The DB2 agent supports the `Info` entry point, which provides static and dynamic information about the database partition and its critical processes. In the example below, the entry point retrieves the database information shown by executing the following commands:

1. Make the configuration writable:

```
# haconf -makerw
```

2. Specify the periodic interval in seconds that the info entry point is invoked. The default value of 0 means info entry point is not invoked.

```
# hatype -modify Db2udb InfoInterval 300
```

3. Show the requested `ResourceInfo` value. In the following example output, the name value pairs for processes monitored by the agent for the DB2 resource.

```
# hares -value db2resource ResourceInfo

State Valid
Msg
PARTITION: 0

UID=db2inst1    PID=21763 PPID=21750 PNAME=db2sysc
```

For more information about the `info` entry point, refer to the *VERITAS Cluster Server User's Guide* and the *VERITAS Cluster Server Agent Developer's Guide*.

Action Entry Point Operation

The DB2 agent supports the `Action` entry point, which enables you to perform predefined actions or custom actions on a resource. To perform an action on a resource, type the following command:

```
# hares -action <res> <token> [-actionargs <arg1> ...] [-sys
<system>] [-rclus <cluster>]
```

The agent supports the following predefined actions:

Predefined Action Token	**Description**
VRTS_GetInstanceName	Retrieves the DB2 instance name of the configured Db2udb resource.
VRTS_GetRunningServices	Retrieves the list of processes monitored by the agent for the Db2udb resource.

For example:

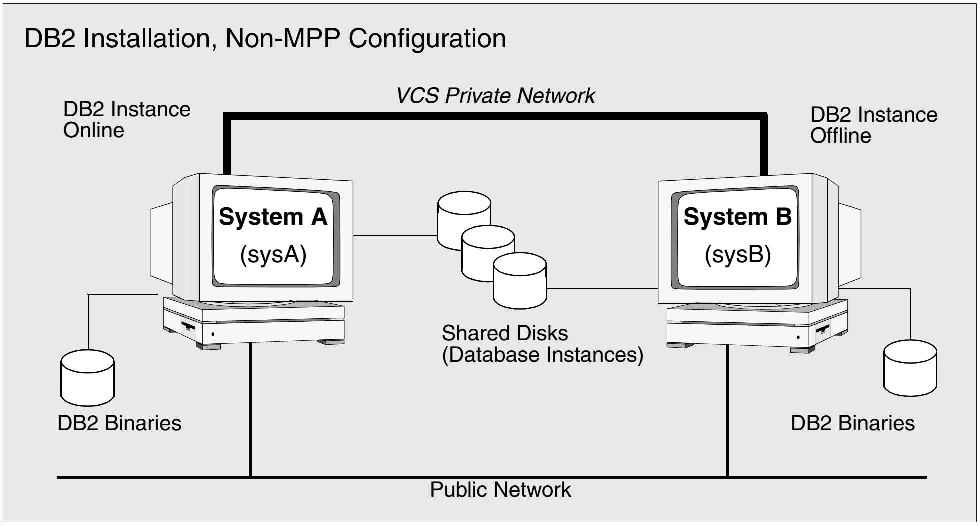
```
# hares -action db2udb0 VRTS_GetInstanceName
VCS NOTICE V-16-13323 Resource (db2udb0): action
(VRTS_GetInstanceName) completed successfully. Output is:

db2inst1

# hares -action db2udb0 VRTS_GetRunningServices

VCS NOTICE V-16-13323 Resource (db2udb0): action
(VRTS_GetRunningServices) completed successfully. Output is:
PARTITION: 0

UID=db2inst1    PID=21763 PPID=21750 PNAME=db2sysc
```

You can also add custom actions for the agent. Refer to the *VERITAS Cluster Server Agent Developer's Guide* for information on defining custom action tokens.

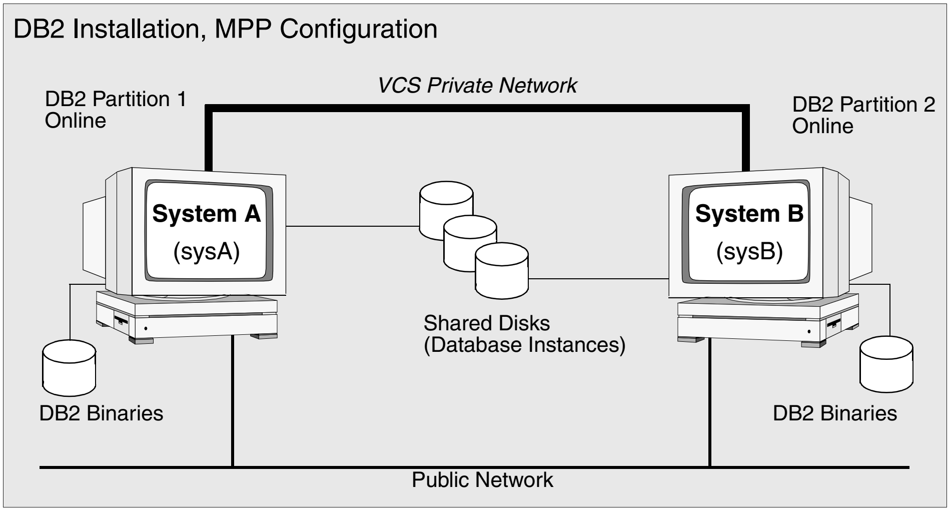
Preparation for Installing DB2 UDB in a VCS Cluster

2

In the following examples, VCS is configured on a two-system cluster. DB2 UDB system binaries are installed identically on local file systems on System A and System B. The instance home directory, instance binaries, and the database reside on shared storage, available to either node. In the case of the non-MPP configuration, an instance is online on only one system at a time, while the other system is a failover system.



In the case of the MPP configuration, a database partition can run on each system and each system can become a failover system.



Prerequisites for Installing DB2 UDB, Non-MPP Versions

- ✔ Verify all systems have enough resources, such as shared memory, to run DB2 UDB. Check the DB2 memory requirements, which vary depending on the version and hardware configuration of DB2. The DB2 UDB system binaries are to be installed locally and the DB2 UDB database instances are to be installed on shared storage.
- ✔ Install and configure VCS version 4.0 or later. Refer to the *VERITAS Cluster Server Installation Guide* for instructions on running either the VERITAS installer or the `installvcs` utility.
- ✔ Before installing DB2 UDB, define DB2 UDB user and group accounts. See "Defining DB2 User and Group Accounts" on page 14. Refer also to the relevant *DB2 Universal Database Quick Beginnings* guide.

Prerequisites for Installing DB2 UDB, MPP Version

- ✔ Verify all systems have enough resources, such as shared memory, to run DB2 UDB. Check the DB2 memory requirements, which vary depending on the version and hardware configuration of DB2. The DB2 UDB system binaries are to be installed on the local file systems on each system and the DB2 UDB database instances are to be installed on shared storage.
- ✔ The MPP configuration requires the Storage Foundation Cluster File System software. This software includes the cluster file system components required by the VCS enterprise agent 4.0 for DB2 UDB, and includes VERITAS Cluster Server (VCS), VERITAS Volume Manager with cluster functionality enabled (CVM), and VERITAS File System with cluster functionality enabled (CFS).

  Refer to the *VERITAS Storage Foundation Cluster File System Installation and Administration Guide* for detailed information on these products and instructions on running either the VERITAS installer or the `installsfcfs` utility.
- ✔ Before installing DB2 UDB, define DB2 UDB user and group accounts. See "Defining DB2 User and Group Accounts" on page 14. Refer also to the relevant *DB2 Universal Database Quick Beginnings* guide.

Creating File Systems for DB2 Instances

The following sections describe examples of creating disk groups for the DB2 database instances.

Creating the File System for the DB2 Non-MPP Instances

To create a file system, you must first create a disk group on the physically shared disk, and create a volume of sufficient size within the disk group. For more information about devices—including shared disks—in the device configuration database, refer to the *VERITAS Cluster Server Installation Guide*.

1. Create a disk group on the shared disk. List the disks using the `lsdev -Cc disk` command. In this case the group consists of one disk, `hdisk5`. For example:

   ```
   # vxdg init db2db_dg hdisk5
   ```

   Deport and import the disk group:

   ```
   # vxdg deport db2db_dg
   # vxdg import db2db_dg
   ```

2. Create a volume of three GB using the vxassist command:

   ```
   # vxassist -g db2db_dg make db2db_vol 3g
   ```

3. Create the file system:

   ```
   # mkfs -V vxfs -o largefiles /dev/vx/dsk/db2db_dg/db2db_vol
   ```

4. Create the mount point directory and mount the file system:

   ```
   # mkdir /db2_mnt/db2inst1
   # mount -V vxfs -o cluster /dev/vx/dsk/db2db_dg/dbq2db_vol
      /db2_mnt/db2inst1
   ```

Creating the Shared Cluster File System for the DB2 MPP Instances

To create a shared file system, you must first create a shared disk group on the physically shared disk and create a volume of sufficient size within the disk group. You must have installed the Storage Foundation Cluster File System software.

1. You must issue the commands to create a shared disk group from the CVM master node. To determine whether a node is the master or the slave, enter the command:

   ```
   # vxdctl -c mode
   ```

   In the output, look for:

   ```
   mode: enabled: cluster active - MASTER
   master: SYSA
   ```

   or:

   ```
   mode: enabled: cluster active - SLAVE
   master: SYSA
   ```

2. From the master node, create the disk group. List the disks using the `vxdisk list` command.

3. Create a shared disk group. In this case the group consists of one disk, `hdisk5`. For example:

   ```
   # vxdg -s init db2db_dg hdisk5
   ```

   Deport and import the disk group:

   ```
   # vxdg deport db2db_dg
   # vxdg -s import db2db_dg
   ```

4. Create a volume of seven GB using the vxassist command:

   ```
   # vxassist -g db2db_dg make db2db_vol 7g
   ```

5. Create the file system:

   ```
   # mkfs -V vxfs -o largefiles /dev/vx/rdsk/db2db_dg/db2db_vol
   ```

6. Create the mount point directory and mount the file system:

   ```
   # mkdir /db2_mnt/db2inst1
   # mount -V vxfs -o cluster /dev/vx/dsk/db2db_dg/dbq2db_vol
     /db2_mnt/db2inst1
   ```

Defining DB2 User and Group Accounts

Before installing DB2 UDB binaries and creating instances, you must define DB2 UDB user and group accounts for each instance on each system. Note the following requirements:

- ✔ The IDs for DB2 users and groups must be exactly the same across all cluster systems.
- ✔ All DB2 user accounts must exist on the local systems. The use of NIS or NIS+ for users is not recommended because these services are not highly available. If their service is interrupted, VCS may not be able to work correctly.

Creating Groups

Three user group accounts are required. Create the group accounts on *each* node in the cluster.

1. Create a group for the DB2 UDB instance owner. For example, enter:

   ```
   # mkgroup id:999 db2iadm1
   ```

2. Create a group for the user to execute fenced user-defined functions (UDFs) or store procedures. For example, enter:

   ```
   # mkgroup id:998 db2fadm1
   ```

3. Create a group for the database administration server. For example, enter:

   ```
   # mkgroup id:997 db2asgrp
   ```

Adding User Accounts

In the following examples that show creating user accounts, the `-g` option specifies the group, `-u` specifies the user ID, `-d` the home directory, `-m` specifies that the home directory is to be created if it doesn't exist, `-s` is the user's login shell, and the final expression is the user's login.

Create the user accounts on *each* node in the cluster.

1. The first example shows creating the user, `db2inst1`, the DB2 UDB instance owner. The mount point, `/db2_mnt/db2inst1`, is used for a file system that hosts the DB2 UDB instance home directory on shared storage, accessible to each node. For example:

   ```
   # mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2.mnt/db2inst1
       db2inst1
   ```

2. The next examples show creating user accounts for `db2fenc1` and `db2as`. These users' home directories are under `/home` in the local file system on each node.

```
# mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/home/db2fenc1
    db2fenc1
# mkuser id=1002 pgrp=db2asgrp groups=db2asgrp home=/home/db2as
    db2as
```

Installing DB2 UDB in VCS Environment

For installing DB2 UDB version 7.x or 8.1 ESE in a VCS environment, we recommend you follow the installation procedure documented in the relevant *DB2 Universal Database Quick Beginnings* guide. Install binaries on local disks of each node, and the database instances on shared storage, accessible by each cluster node.

Setting Shared Memory Parameters

For details on setting the shared memory parameters, refer to the relevant *DB2 Universal Database Quick Beginnings* guide.

Installing the Binaries

Install the DB2 UDB system binaries on local disks on each node (mirrored disks are recommended), not on shared storage. You can use the `db2setup` tool.

Install the DB2 License

Install the DB2 license on each node. For example, enter:

```
# /usr/opt/db2_08.01/adm/db2licm -a db2ese.lic
```

Installing the Instances

Install the database instances on the shared storage only on the one node where the instance's home directory is currently mounted. You can choose to install single-partition instance or multi-partition instance. You can use the `db2setup` tool.

- ✔ When using `db2setup`, do not select the option to "`Auto start DB2 instance at system boot`" in the DB2 Instance Properties window (if this option exists for your DB2 version). VCS needs to bring up the resources for the DB2 instances in a specific order before bringing the instance itself online.
- ✔ The instance's home directory is a mount point on the shared storage.

Setting Up the DB2 UDB Configuration

Use the following procedures to configure DB2 UDB ESE multi-partition instance (non-MPP) and DB2 UDB ESE multi-partition instance (MPP) in a VCS environment.

Checking /etc/services

On each system in the cluster, check the file `/etc/services`; use the `more` command.

- ✔ Make sure each partition has a port number assigned. The number of ports reserved depends on the number of partitions.
- ✔ Make sure the ports are not used by any other services. Manually assign new numbers if necessary.
- ✔ Make sure all systems in the cluster have the same entries in the `/etc/services` file.

The following is an example for a DB2 UDB instance with two partitions, one port for each partition:

```
# more /etc/services
DB2_db2inst1      70000/tcp # Connection port for DB2 instance
   db2inst1
DB2_db2inst1_END  70001/tcp
DB2_db2inst1a     60003/tcp # Interrupt port for DB2 instance
   db2inst1
DB2_db2inst10     50002/tcp # Connection port for DB2 instance
   db2inst1
DB2_db2inst10     50003/tcp # Interrupt port for DB2 instance
   db2inst1
```

The file shows the ports assigned when DB2 UDB creates a new instance. The first two lines in the example show a range of ports, 70000 through 70001, for two database partitions. The next three lines assign additional ports for use by the database instance. Inspect the file and verify there are no duplicate port numbers.

The following is an example for a DB2UDB instance with four partitions, a port for each partition.

```
# more /etc/services
DB2_db2inst1     60000/tcp #
DB2_db2inst1_1   60001/tcp #
DB2_db2inst1_2   60002/tcp #
DB2_db2inst1_END 60003/tcp #
db2c_db2inst1    50000/tcp #
```

The four instances are assigned 60000 through 60003.

Creating $DB2InstHome/.rhosts

On each system, create a file named `$DB2InstHome/.rhosts`, and place a “+” character within it. This file permits a system to access the database without being prompted for a password.

If security is a concern, put the `hostname` and `userid` inside the `.rhosts` file, as shown in the following examples:

```
dbmach01   db2inst1
dbmach02   db2inst1
dbmach03   db2inst1
dbmach04   db2inst1
```

or,

```
+   db2inst1
```

Using the command, `rsh` *`system_name`*, test that you can remotely log in with the DB2 instance (for example, `db2inst1)` account from one system in the cluster to another without being prompted for a password. Test this from each system in the cluster to all other systems.

Modifying the $DB2InstHome/sqllib/db2nodes.cfg File

The `$DB2InstHome/sqllib/db2nodes.cfg` file is used by DB2 during failover from one node to another.

Non-MPP Versions

For each DB2 UDB ESE multi-partition instance (non-MPP) database partitions, modify the file `$DB2InstHome/sqllib/db2nodes.cfg` such that you create an entry for each database partition, assigning the virtual IP address as the hostname. For example:

```
0 virtualhostname 0
1 virtualhostname 1
```

Note that the *virtualhostname* corresponds to the virtual IP address listed in the file `/etc/hosts`. Make sure that the virtual IP address is up and running at this time.

MPP Versions

For MPP versions, modify the file `$DB2InstHome/sqllib/db2nodes.cfg` with the hostname that you want each database partition to start on. The `db2nodes.cfg` file is automatically changed and updated by DB2 to enable the database partitions to fail over from one node to another. DB2 adds a fourth column for the "netname," which is, by default, the hostname. The virtual IP is not used in the `db2nodes.cfg` file for MPP configurations.

As a two-node sample:

```
0 sysA 0
1 sysB 0
```

Make sure that the relative port number in the third column is unique for each partition on a host. For example:

```
0 sysA 0
1 sysA 1
2 sysB 0
3 sysB 1
```

Confirming the Setup of DB2 MPP Installation

On the host where the shared file system is mounted, check whether you can start and stop each instance, and thus verify that DB2 is properly installed.

1. Log in as the instance owner:

   ```
   # su - db2inst1
   ```

2. Attempt to start the instance:

   ```
   $ db2start
   ```

   The application should start on the nodes specified in the `db2nodes.cfg` file.

3. Assuming the previous command is successful, stop the instance:

   ```
   $ db2stop
   ```

4. If the application does not start successfully or stop correctly on each node, check for configuration errors. Review your DB2 UDB documentation, such as the *DB2 Universal Database Quick Beginnings Guide*.

5. Create a database.

   ```
   $ db2 create database dbname
   ```

6. List the database directory

   ```
   $ db2 list database directory
   ```

Installing the DB2 UDB Agent Software 3

Use the following procedure to install the DB2 UDB agent software.

▼ To install the software

1. Log in as `root`.

2. Determine the device access name of the CD-ROM drive. For example, enter:

   ```
   # cd /dev
   # lsdev -C -c cdrom
   ```

3. Insert the VERITAS software disc containing the DB2 UDB enterprise agent software into a CD-ROM drive connected to your system.

4. Mount the CD using the device access name found in step 2. For example:

   ```
   # mkdir -p /cdrom
   # mount -V cdrfs -o ro /dev/cd0 /cdrom
   ```

5. Add the fileset for the DB2 UDB enterprise agent:

   ```
   # cd /cdrom
   # installp -a -d /cdrom/pkgs/VRTSvcsdb.rte.bff VRTSvcsdb
   ```

   If you do *not* require Japanese language support, repeat step 1 through step 5 on each system in the cluster to complete installation of the agent software.

6. (Optional) If you require Japanese language support, install the Japanese language message catalogue package:

   ```
   # installp -ax -d . VRTSjacsb
   ```

   Repeat step 1 through step 6 on each system in the cluster to complete installation of the agent software.

Configuring the DB2 Agent 4

This chapter describes how you can configure the DB2 UDB agent. You can configure the agent using three methods:

◆ By using VCS Cluster Manager (the Java Console) to edit a service group template for the DB2 UDB agent. See "Configuring the DB2 UDB Agent Using Cluster Manager" on page 26.

◆ By using VCS commands. Refer to the *VERITAS Cluster Server User's Guide* for information about configuring VCS from the command line.

◆ By editing the main.cf file directly, using the types configuration file and referring to the sample `main.cf` file supplied with the DB2 UDB agent (see "Configuring the DB2 UDB Agent by Editing the main.cf File" on page 32). This method requires that VCS be stopped and restarted before the configuration takes effect.

Configuring the Agent

Configuring the DB2 UDB agent involves assigning values to the DB2 UDB resource type attributes, which are described in the following table for your review and reference. The resource type definition file, `Db2udbTypes.cf`, is also shown for reference. The sample `main.cf` configuration files are shown in "Sample Configuration Files" on page 41.

Db2udb Resource Type Attributes

The required and optional attributes are described below. The temporary attribute, `LastWarningDay` (not described below), is internally used by the agent to ensure that the same error messages are not repeatedly logged.

Required Attribute	**Type and Dimension**	**Definition**
DB2InstOwner	string-scalar	User ID of Instance Owner that starts a DB2 UDB instance. Each instance requires a unique user ID.
DB2InstHome	string-scalar	Path to DB2 UDB instance home directory that contains configuration files for the DB2 instance.

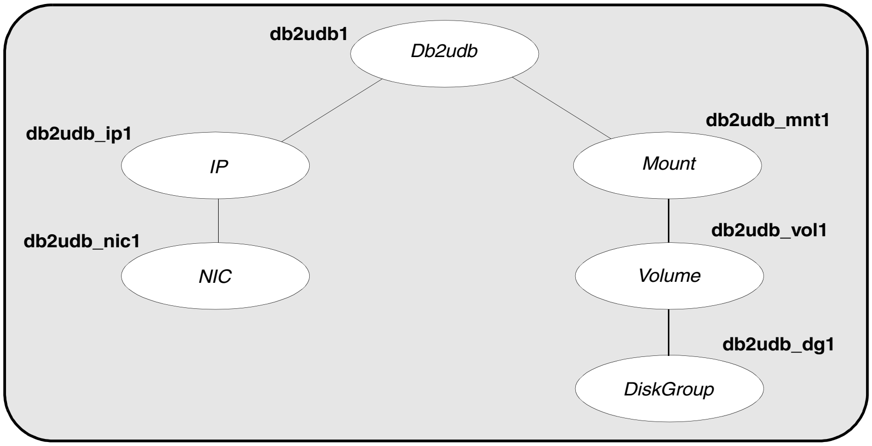
Optional Attribute	**Type and Dimension**	**Definition**
IndepthMonitor	integer-scalar	Flag indicating if in-depth monitor is enabled (1) or disabled (0).The default is 0.
DatabaseName	string-scalar	Name of the database for in-depth monitoring; required if in-depth monitor is enabled (`IndepthMonitor = 1`).
NodeNumber	integer-scalar	Node number or partition number of the database. Used when monitoring a specific database partition in ESE multi-partition instance environment. Default value is 0 for ESE single-partition instance and multi-partition instance configurations.
Encoding	string-scalar	Specifies operating system encoding corresponding to DB2 UDB encoding for display of DB2 UDB output. For example, if the environment variable LANG is set to "ja_JP," then "IBM-`eucJP`" is the AIX value for `Encoding`. Refer to DB2 UDB documentation for encoding values. The default is "".
WarnOnlyIfDBQueryFailed	boolean-scalar	Flag indicating if the DB2 resources should be faulted when the in-depth monitor fails. In-depth monitoring consists of a database query. ◆ If the attribute is set to 0, the agent faults the DB2 resource if query fails. ◆ If the attribute is set to 1, the agent issues a warning message about the query failure; the resource remains online.

DB2 UDB Type Definition File, Db2udbTypes.cf

```
type Db2udb (
     static keylist SupportedActions = {VRTS_GetInstanceName,
        VRTS_GetRunningServices }
     static int CleanTimeout = 240
     static int MonitorTimeout = 240
     static int OfflineTimeout = 240
     static int OnlineRetryLimit = 2
     static int OnlineTimeout = 180
     static int OnlineWaitLimit = 1
     static int RestartLimit = 1
     static int ToleranceLimit = 1
     static str ArgList[] = { DB2InstOwner, DB2InstHome,
        IndepthMonitor, DatabaseName, NodeNumber, Encoding,
        WarnOnlyIfDBQueryFailed, LastWarningDay }
     str DB2InstOwner
     str DB2InstHome
     int IndepthMonitor
     str DatabaseName
     int NodeNumber
     str Encoding
     boolean WarnOnlyIfDBQueryFailed = 1
     temp str LastWarningDay
)
```

DB2 UDB Service Group for Non-MPP Configuration

The figure below illustrates the dependencies among the resources configured for a non-MPP DB2 UDB instance service group.

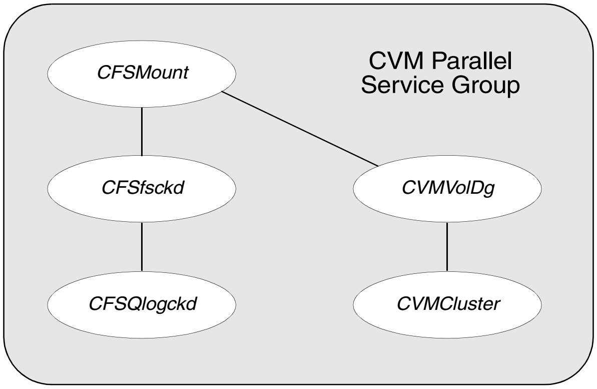


This configuration shows a resource group for an instance of DB2 UDB. The `db2udb1` resource (the database) requires the IP resource and the Mount resource. The service group IP address for the DB2 UDB server is configured using the IP resource (`db2udb_ip1`) and the NIC resource (`db2udb_nic1`). The mount resource (`db2udb_mnt1`) requires the Volume resource (`db2udb_vol1`) which in turn requires the DiskGroup resource (`db2udb_dg1`). The DB2 UDB instance can be started after each of these resources is available.

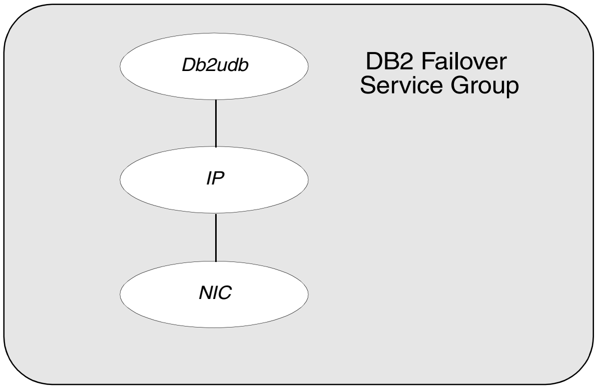
DB2 UDB Service Groups for MPP Configuration

The DB2 UDB agent uses two service groups to support MPP configuration for DB2 version 8.1 and above.

The first service group is a parallel CVM service group. There is one CVM/Infrastructure group per cluster node. This service group has the CVM resource and the necessary resources for support of CFS. This group also contains all common components needed by DB2, such as the instance's home directory, which is shared on all the cluster nodes.



The second service group is a failover DB2 service group. This service group monitors one database partition for DB2 version 8 with MPP configuration. The failover DB2 service group depends on the parallel CVM service group with online local firm dependency.



Configuring the DB2 UDB Agent Using Cluster Manager

Templates for the DB2 UDB service groups were automatically installed when you installed the DB2 UDB enterprise agent. Templates include one for the DB2 UDB MPP configuration and one for the DB2 UDB non-MPP configuration. Using the VCS Cluster Manager (Java Console), you can use a template to configure the DB2 UDB service group, its resources, and their attributes. You can also use the Java Console to dynamically modify the attributes' values as necessary for your configuration.

Refer to the *VERITAS Cluster Server User's Guide* for information on the VCS Java Console.

Importing the Db2udbTypes.cf File

To use the DB2 UDB templates, import the `Db2udbTypes.cf` file to the VCS engine by using Cluster Manager (Java Console):

1. On one of the systems of the cluster, start Cluster Manager.

```
# hagui&
```

2. Log into the cluster and wait for Cluster Explorer to launch.

3. In the Cluster Explorer window, click File and select Import Types from the drop down menu. Switch to the read/write mode if prompted.

4. In the Import Type dialog box, enter the pathname for the `Db2udbTypes.cf` file in the File Name box:

```
/etc/VRTSvcs/conf/config/Db2udbTypes.cf
```

5. Click **Import** and wait for the file to be imported.

6. In the Cluster Explorer window, click the **Save Configuration** icon.

   When the DB2 UDB types are imported to the VCS engine, the DB2 UDB agent can be configured.

   - If you are using the DB2 UDB MPP configuration, go to "Adding the Service Group for DB2 UDB MPP" on page 27.
   - If you are using the DB2 UDB non-MPP configuration, go to "Adding the Service Group for DB2 UDB Non-MPP" on page 29.

Adding the Service Group for DB2 UDB MPP

If you have imported the `Db2udbTypes.cf` file (see "Importing the Db2udbTypes.cf File" on page 26), you can use the Db2udb_MPP_Group template to configure a service group.

After you log into Cluster Manager, the Status tab should indicate that the CVM service group is Online on each system in the cluster. The CVM service group is automatically configured when you complete the installation of the Storage Foundation Cluster File System (SFCFS) software.

To add the service group for the DB2 UDB MPP database, do the following:

1. In the Cluster Explorer window, click the **Add Service Group** icon on the toolbar.

2. In the Add Service Group window, do the following:

    a. Enter the name of the service group you want to create for the service group. For example, enter `db2mpp_grp1`. Do not press Return or Enter.

    b. From the systems shown in the Available Systems box, double-click those you want in your configuration.

    c. In the window showing the systems added to the configuration, click the checkbox for system on which you want to automatically start the service group.

    d. Click the **Failover** radio button to specify the Service Group Type.

    e. Click the **Templates** button.

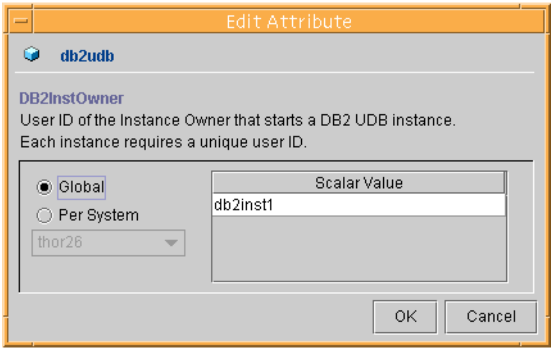
    f. In the Select Templates window, select **`db2udb_mpp_grp`** from the list shown in the Templates box. The Dependency graph information and the Types information should change to reflect the template choice. Click **OK**. In the Add Service Group window, the name of the template is now shown as selected.

    g. Click **OK** on at the bottom of the Add Service Group window. The group is added. On the left pane of the Cluster Manager window, the service group is shown below the CVM service group. On the Status tab, the group is shown Offline on each system.

3. In the left pane, double-click the **`db2mpp_grp1`** service group. The types of resources that can be configured for the group are displayed: Db2udb, IP, and NIC.

4. Double-click the **`Db2udb`** resource type. Select the resource, **`db2udb`**, below the Db2udb type and click the **Properties** tab.

5. On the Properties tab for the `db2udb` resource, a list of Type Specific Attributes is shown. Click the **Edit** icon for each attribute you want to configure. In the Edit Attribute window, enter the necessary attribute value information. For example, enter the **`db2inst1`** as the value for `DB2InstOwner`.

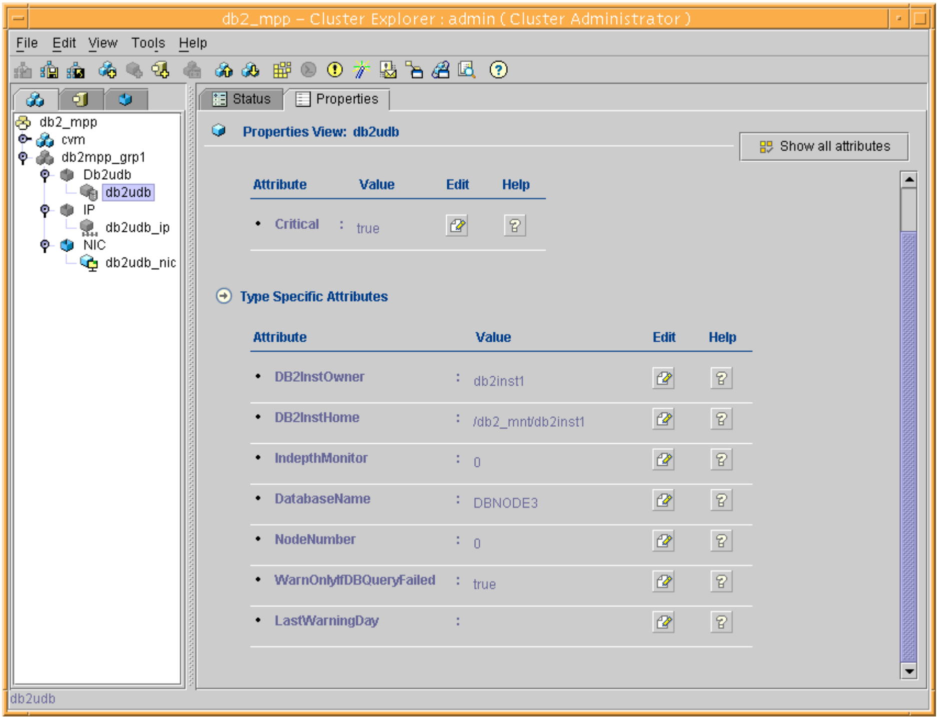


DB2InstOwner
User ID of the Instance Owner that starts a DB2 UDB instance.
Each instance requires a unique user ID.

● Global
○ Per System
thor26

Scalar Value
db2inst1

After you have assigned the attribute values, the list of Type Specific Attributes resembles the following:



Properties View: db2udb

Attribute	Value	Edit	Help
Critical	: true		

Type Specific Attributes

Attribute	Value	Edit	Help
DB2InstOwner	: db2inst1		
DB2InstHome	: /db2_mnt/db2inst1		
IndepthMonitor	: 0		
DatabaseName	: DBNODE3		
NodeNumber	: 0		
WarnOnlyIfDBQueryFailed	: true		
LastWarningDay	:		

6. Assign values for the IP and NIC resources in the same manner as you assigned values to the db2udb resource: double-click the type to display the resource and select the resource. With the Properties tab visible, you can edit the Type Specific Attributes for each resource.

7. Right-click the **`db2mpp_grp1`** service group in the left pane. Click **Link** in the drop-down menu.

   The Link Service Groups window shows the Parent Group as db2mpp_grp1, the Child group as CVM, the Relationship as “online local,” and the Dependency Type as “firm.”

8. Click **OK** to create the dependency link.

9. Click the icon for **Save Configuratio**n.

10. Enable the `db2udb` and `IP` resources. Right-click a resource and select **Enabled** in the drop-down menu. If necessary, make the configuration read/write.

11. Click the **Online Service Group** icon.

12. In the window, select the service group and the system on which you want to online. Click **OK**.

Adding the Service Group for DB2 UDB Non-MPP

If you have imported the `Db2udbTypes.cf` file (see “Importing the Db2udbTypes.cf File” on page 26), you can use the Db2udb_Group template to configure a service group.

1. In the Cluster Explorer window, answer **No** when prompted to use the configuration wizard. Note, if you choose to use the wizard, the steps that follow are similar.

2. In the Cluster Explorer window, click the **Add Service Group** icon on the toolbar.

3. In the Add Service Group window, do the following:

   a. Enter the name of the service group you want to create for the service group. For example, enter **`db2_group1`**. Do not press Return or Enter.

   b. From the systems shown in the Available Systems box, double-click those you want in your configuration.

   c. In the window showing the systems added to the configuration, click the checkbox for a system on which you want to automatically start the service group.

d. Click the **Failover** radio button to specify the Service Group Type.

e. Click the **Templates** button.

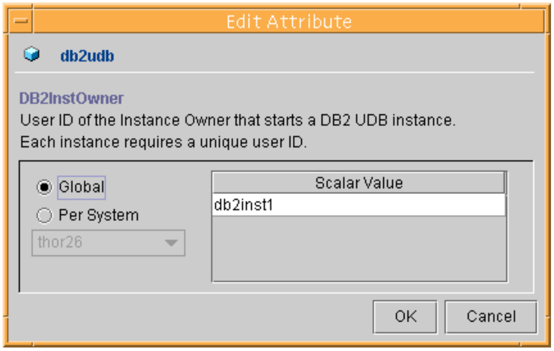
f. In the Select Templates window, select **`db2udb_grp`** from the list shown in the Templates box. The Dependency graph information and the Types information should change to reflect the template choice. Click **OK**. In the Add Service Group window, the name of the template is now shown as selected.

g. Click **OK** on at the bottom of the Add Service Group window. The group is added. On the left pane of the Cluster Manager window, the service group is shown below the CVM service group. On the Status tab, the group is shown Offline on each system.

4. In the left pane, double-click the **`db2_group1`** service group. The types of resources that can be configured for the group are displayed: Db2udb, DiskGroup, IP, Mount, NIC, and Volume.

5. Double-click the **`Db2udb`** resource type. Select the resource, **`db2udb`**, below the Db2udb type and click the **Properties** tab.

6. On the Properties tab for the `db2udb` resource, a list of Type Specific Attributes is shown. Click the **Edit** icon for each attribute you want to configure. In the Edit Attribute window, enter the necessary attribute value information. For example, enter the `db2inst1` as the value for `DB2InstOwner`.



Edit Attribute

db2udb

DB2InstOwner

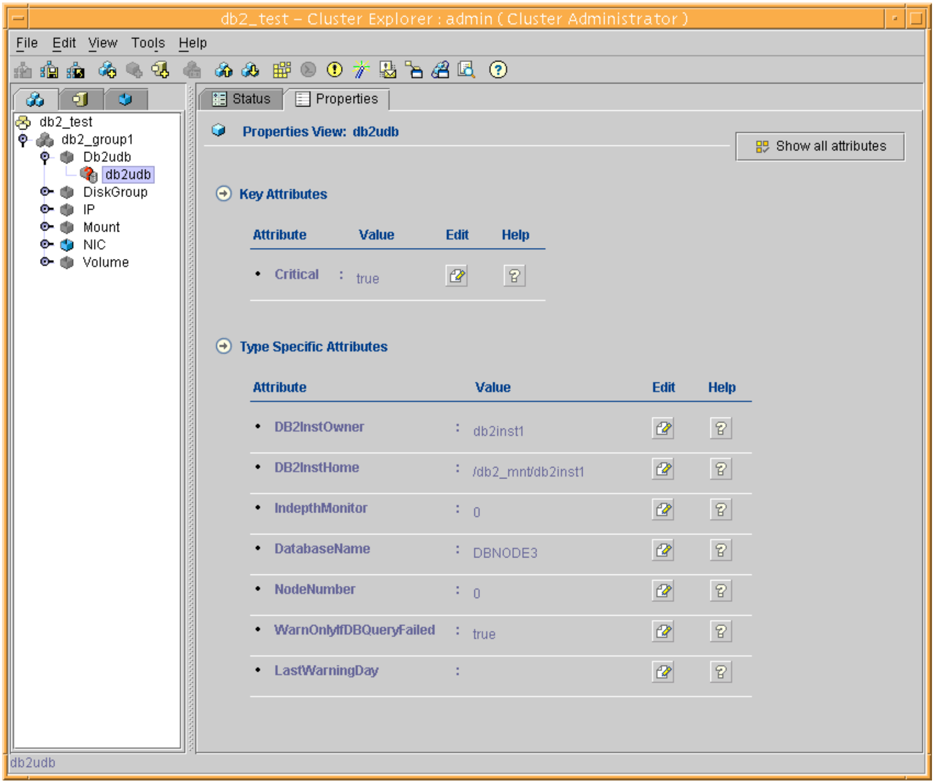
User ID of the Instance Owner that starts a DB2 UDB instance.
Each instance requires a unique user ID.

● Global

○ Per System

Scalar Value
db2inst1

After you have assigned the attribute values, the list of Type Specific Attributes resembles the following:



Key Attributes

Attribute	Value	Edit	Help
Critical	true		

Type Specific Attributes

Attribute	Value	Edit	Help
DB2InstOwner	db2inst1		
DB2InstHome	/db2_mnt/db2inst1		
IndepthMonitor	0		
DatabaseName	DBNODE3		
NodeNumber	0		
WarnOnlyIfDBQueryFailed	true		
LastWarningDay			

7. Assign values for the DiskGroup, IP, Mount, NIC, and Volume resources in the same manner as you assigned values to the db2udb resource: double-click the type to display the resource and select the resource. With the Properties tab visible, you can edit the Type Specific Attributes.

8. Click the icon for **Save and Close Configuration**.

9. Enable the resources in db2_group1. Right-click each resource and select Enabled in the drop-down menu. If necessary, make the configuration read/write.

10. Click the Online Service Group icon.

11. In the window, select the service group and the system on which you want to online. Click **OK**.

Configuring the DB2 UDB Agent by Editing the main.cf File

The VCS enterprise agent for DB2 UDB comes with three sample VCS configuration files installed in the `/etc/VRTSvcs/conf/sample_Db2udb` directory. One sample is for an ESE single-partition instance configuration, another for a ESE multi-partition instance SMP configuration, and another for an ESE multi-partition instance MPP configuration. The appropriate file can be used as reference to directly modify your present `main.cf` configuration file. When you use this method, you must stop and restart VCS to implement the configuration.

Preparation for Editing the main.cf File

1. Log in to System A as `root`.

2. Save your existing configuration to prevent any changes while you modify the `main.cf` file:

```
# haconf -dump -makero
```

3. Ensure VCS is not running while you edit `main.cf` by using the `hastop` command to stop the VCS engine on all systems and leave the resources available:

```
# hastop -all -force
```

4. Make a backup copy of the `main.cf` file:

```
# cd /etc/VRTSvcs/conf/config
# cp main.cf main.cf.orig
```

Depending on your configuration, go to one of the following sections that describe configuring the DB2 agent.

Configuring the Agent to Use the MPP Configuration

Edit the `main.cf` file. Use `/etc/VRTSvcs/conf/sample_Db2udb/main.cf.MPP` for reference. Notice that CVM service group is present in the configuration file.

1. Add an “include” line for the `Db2udbTypes.cf` file:

```
include "Db2udbTypes.cf"
```

2. Create service groups for the DB2 UDB resources. Refer to the sample configuration file “Sample MPP Configuration: DB2 UDB ESE Multi-Partition Instance” on page 47. The example shows four DB2 MPP service groups and a CVM service group.

3. In the DB2 MPP service groups, Include the definitions for the Db2udb, IP, and NIC resources, and assign values to the attributes for the resources to match the parameters of your configuration.

   Refer to the "Db2udb Resource Type Attributes" on page 22 as well as the sample configuration files. Refer also to the *VERITAS Cluster Server Bundled Agents Reference Guide* for information about IP and NIC resources.

4. Assign the `online local firm` service group dependency of the db2udb service group for the cvm service group. For example:

   ```
   requires group cvm online local firm
   ```

5. Immediately following the service group dependency, assign dependencies for the newly created resources. Refer to the appropriate sample configuration file. (See the *VCS User's Guide* for more information on assigning dependencies.) For example, referring to the "Sample MPP Configuration: DB2 UDB ESE Multi-Partition Instance" on page 47, for the group `db2mpp_grp0` you would enter:

   ```
   db2udb0 requires Db2_IP0
   Db2_IP0 requires Db2_NIC0
   ```

6. Save and close the file.

Configuring the Agent to Use Non-MPP Configurations

Edit the `main.cf` file. Use `/etc/VRTSvcs/conf/sample_Db2udb/main.cf.EE` or `/etc/VRTSvcs/conf/sample_Db2udb/main.cf.EEE` for reference:

1. Add an "include" line for the `Db2udbTypes.cf` file:

   ```
   include "Db2udbTypes.cf"
   ```

2. Create a service group for the DB2 UDB resources.

   If you are using DB2 UDB ESE single-partition instance, refer to the example, "Sample Non-MPP Configuration: DB2 UDB ESE Single-Partition Instance" on page 41 which shows two groups, "db2udb_grp1," and "db2udb_grp2."

   If you are using DB2 UDB ESE multi-partition instance, refer to the example, "Sample Non-MPP Configuration: DB2 UDB ESE Multi-Partition Instance" on page 45, which shows a group named "db2_grp1" in which two partitions are defined.

3. Include all resources in the service groups, including the Db2udb, DiskGroup, IP, Mount, NIC, and Volume resources, and assign values to the attributes for the resources to match the parameters of your configuration.

   Refer to the “Db2udb Resource Type Attributes” on page 22 as well as the sample configuration files. Refer also to the *VERITAS Cluster Server Bundled Agents Reference Guide* for information about the DiskGroup, IP, Mount, NIC, and Volume resources.

4. Assign dependencies for the newly created resources. Refer to the appropriate sample configuration file. (See the *VCS User's Guide* for more information on assigning dependencies.) For example, referring to the “Sample Non-MPP Configuration: DB2 UDB ESE Single-Partition Instance” on page 41, for the group db2udb_grp1 you would enter:

   ```
   db2udb1 requires db2udb_ip1
   db2udb1 requires db2udb_mnt1
   db2udb_ip1 requires db2udb_nic1
   db2udb_mnt1 requires db2udb_vol1
   db2udb_vol1 requires db2udb_dg1
   ```

   And for group db2udb_grp3 you would enter:

   ```
   db2udb3 requires db2udb_ip3
   db2udb3 requires db2udb_mnt3
   db2udb_ip3 requires db2udb_nic3
   db2udb_mnt3 requires db2udb_vol3
   db2udb_vol3 requires db2udb_dg3
   ```

5. Save and close the file.

Verifying the Configuration

After editing the `main.cf` file for you configuration, check the configuration using the following procedure:

1. Copy the DB2 UDB types configuration file into place:

   ```
   # cp /etc/VRTSvcs/conf/Db2udbTypes.cf
       /etc/VRTSvcs/conf/config/Db2udbTypes.cf
   ```

2. Verify the syntax of the file `/etc/VRTSvcs/conf/config/main.cf`:

   ```
   # cd /etc/VRTSvcs/conf/config
   # hacf -verify .
   ```

3. Start the VCS engine on System A:

   ```
   # hastart
   ```

4. Type the command `hastatus`:

   ```
   # hastatus
   ```

5. When “`LOCAL_BUILD`” is listed in the message column, start VCS on System B:

   ```
   # hastart
   ```

6. Verify that all DB2 UDB service group resources are brought online on System A:

   ```
   # hagrp -display
   ```

7. Take the service groups offline on System A and verify that all resources are stopped:

   ```
   # hagrp -offline db2udb_grp1 -sys sysa
   # hagrp -offline db2udb_grp3 -sys sysa
   # hagrp -display
   ```

8. Bring the service groups online again on System A and verify that all resources are available:

   ```
   # hagrp -online db2udb_grp1 -sys sysa
   # hagrp -online db2udb_grp3 -sys sysa
   # hagrp -display
   ```

9. Switch the DB2 UDB service group to System B:

   ```
   # hagrp -switch db2udb_grp1 -to sysb
   # hagrp -switch db2udb_grp3 -to sysb
   ```

10. Verify that all DB2 UDB service group resources are brought online on System B:

    ```
    # hagrp -display
    ```

11. On all the systems, look at the following log files for any errors or status:

    ```
    /var/VRTSvcs/log/engine_A.log
    /var/VRTSvcs/log/Db2udb_A.log
    ```

Modifying the Agent Configuration

To dynamically reconfigure the VCS enterprise agent for DB2 UDB, use Cluster Manager or the VCS command line. The following description of changing the configuration to include in-depth monitoring shows the use of VCS commands from the command line. See the chapter on reconfiguring VCS from the command line in the *VERITAS Cluster Server User's Guide.*

Enabling In-Depth Monitoring of DB2 UDB Instance

Shallow monitoring of a DB2 UDB instance involves either checking the "`db2nps`" output, which displays active processes for the instance or the database partition, or checking the exit status of the "`db2gcf`" command. By contrast, in-depth monitoring provides a higher level of confidence in the availability of the instance or partition and its database by making additional queries to the database to verify whether the database is available.

Enabling In-Depth Monitoring from the Command LIne

You can dynamically configure in-depth monitoring. It is recommended that you successfully run DB2 UDB with the agent's default (shallow) monitoring before you start the in-depth monitoring. In the MPP configuration, make sure the database can be accessible locally by the database partition.

To start the in-depth monitor for a given instance:

1. Make the VCS configuration writable:

   ```
   # haconf -makerw
   ```

2. Freeze the service group so VCS does not perform actions automatically based on an incomplete reconfiguration:

   ```
   # hagrp -freeze db2udb_grp1
   ```

3. Enable in-depth monitoring using the command:

   ```
   hares -modify resource DatabaseName name
   hares -modify resource IndepthMonitor 1
   ```

   For example:

   ```
   # hares -modify db2udb DatabaseName SAMPLE
   # hares -modify db2udb IndepthMonitor 1
   # haconf -dump -makero
   # hagrp -unfreeze db2udb_grp1
   ```

Disabling In-Depth Monitoring

You can dynamically disable in-depth monitoring as follows:

1. Make the VCS configuration writable:

```
# haconf -makerw
```

2. Freeze the service group so VCS does not perform actions automatically based on an incomplete reconfiguration:

```
# hagrp -freeze db2udb_grp1
```

3. Disable in-depth monitoring by assigning the `MonScript` attribute a null value. Use the command:

```
hares -modify resource IndepthMonitor 0
```

For example:

```
# hares -modify db2udb IndepthMonitor 0
# haconf -dump -makero
# hagrp -unfreeze db2udb_grp1
```

Disabling and Removing the Agent 5

This chapter describes how to disable or remove the DB2 UDB agent.

Disabling the Agent

To disable the agent on a system, you must first change the DB2 UDB service group to an OFFLINE state on the system. You can stop the application completely, or switch the service group to another system.

1. Determine if the service group is online by entering:

   ```
   # hagrp -state service_group -sys system_name
   ```

2. If the service group is online, take it offline by entering:

   ```
   # hagrp -switch service_group -to system_name
   ```

   or:

   ```
   # hagrp -offline service_group -sys system_name
   ```

3. Stop the agent on the system by entering:

   ```
   # haagent -stop service_group -sys system_name
   ```

   When you get the message "`Please look for messages in the log file`," check the file `/var/VRTSvcs/log/engine_A.log` for a message confirming the agent has stopped.

   You can also use the `ps` command to confirm the agent is stopped.

When the agent is stopped, you can remove the system, the service group, or the resource type from the VCS configuration. See the chapter on reconfiguring VCS from the command line in the *VERITAS Cluster Server User's Guide* for more information.

Removing the Agent

Type the following command on each system to remove the agent. Answer prompts accordingly:

```
# installp -u VRTSvcsdb
```

A

Sample Configuration Files

This chapter shows example DB2 UDB configurations in VCS configuration files.

Sample Non-MPP Configuration: DB2 UDB ESE Single-Partition Instance

The following configuration reflects DB2 UDB with two instances configured in a ESE single-partition instance environment.

```
include "types.cf"
include "Db2udbTypes.cf"

cluster db2_clus (
        UserNames = { admin = "cDRpdxPmHpzS." }
        Administrators = { admin }
        CounterInterval = 5
        )

system sysA (
        CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
                ActionTimeLimit = 0, Action = NONE,
                NotifyThreshold = 0, NotifyTimeLimit = 0 }
        )

system sysB (
        CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
                ActionTimeLimit = 0, Action = NONE,
                NotifyThreshold = 0, NotifyTimeLimit = 0 }
        )

group db2udb_grp1 (
        SystemList = { sysA= 0, sysB = 1 }
        AutoStartList = { sysA }
        )
```

```
Db2udb db2udb1 (
        DB2InstOwner = db2inst1
        DB2InstHome = "/db2inst1"
        IndepthMonitor = 1
        DatabaseName = SAMPLE
        )

DiskGroup db2udb_dg1 (
        DiskGroup = db2_dg1
        )

IP db2udb_ip1 (
        Device = en0
        Address = "166.98.9.163"
        NetMask = "255.255.220.0"
        )

Mount db2udb_mnt1 (
        MountPoint = "/db2inst1"
        BlockDevice = "/dev/vx/dsk/db2_dg1/inst1_vol"
        FSType = vxfs
        MountOpt = rw
        )

NIC db2udb_nic1 (
        Device = en0
        NetworkHosts = { "166.98.128.180" }
        )

Volume db2udb_vol1 (
        Volume = inst1_vol
        DiskGroup = db2_dg1
        )

db2udb1 requires db2udb_ip1
db2udb1 requires db2udb_mnt1
db2udb_ip1 requires db2udb_nic1
db2udb_mnt1 requires db2udb_vol1
db2udb_vol1 requires db2udb_dg1
```

```
        // resource dependency tree
        //
        //      group db2udb_grp1
        //      {
        //      Db2udb db2udb1
        //         {
        //         IP db2udb_ip1
        //            {
        //            NIC db2udb_nic1
        //            }
        //         Mount db2udb_mnt1
        //            {
        //            Volume db2udb_vol1
        //                 {
        //                 DiskGroup db2udb_dg1
        //                 }
        //            }
        //         }
        //      }

group db2udb_grp2 (
        SystemList = { sysA = 0, sysB = 1 }
        AutoStartList = { sysA }
        )

        Db2udb db2udb2 (
                DB2InstOwner = db2inst2
                DB2InstHome = "/db2inst2"
                IndepthMonitor = 1
                DatabaseName = MYDB
                )

        IP db2udb_ip2 (
                Device = en0
                Address = "166.98.9.168"
                NetMask = "255.255.220.0"
                )

        LVMVG db2udb_vg00 (
                VolumeGroup = vg00
                MajorNumber = 50
                Disks = { hdisk2, hdisk3 }
                VaryonvgOpt = p
                )
```

```
Mount db2udb_mnt2 (
        MountPoint = "/db2inst2"
        BlockDevice = "/dev/vx/dsk/db2_dg2/inst2_vol"
        FSType = vxfs
        MountOpt = rw
        )

NIC db2udb_nic2 (
        Device = en0
        NetworkHosts = { "166.98.128.181" }
        )

db2udb2 requires db2udb_ip2
db2udb2 requires db2udb_mnt2
db2udb_ip2 requires db2udb_nic2
db2udb_mnt2 requires db2udb_vg00

// resource dependency tree
//
//      group db2udb_grp2
//      {
//      Db2udb db2udb2
//         {
//         IP db2udb_ip2
//              {
//              NIC db2udb_nic2
//              }
//         Mount db2udb_mnt2
//              {
//              LVMVG db2udb_vg00
//              }
//         }
//      }
```

Sample Non-MPP Configuration: DB2 UDB ESE Multi-Partition Instance

The following `main.cf` configuration file reflects DB2 UDB in a ESE multi-partition instance SMP environment. Two database partitions are shown.

```
include "types.cf"
include "Db2udbTypes.cf"

cluster db2_clus (
        UserNames = { admin = "cDRpdxPmHpzS." }
        Administrators = { admin }
        CounterInterval = 5
        )

system sysA (
        CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
                 ActionTimeLimit = 0, Action = NONE,
                 NotifyThreshold = 0, NotifyTimeLimit = 0 }
        )

system sysB (
        CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
                 ActionTimeLimit = 0, Action = NONE,
                 NotifyThreshold = 0, NotifyTimeLimit = 0 }
        )
group db2_grp1 (
        SystemList = { sysA = 0, sysB = 1 }
        AutoStartList = { sysA }
        )

        Db2udb db2udb1 (
                DB2InstOwner = db2inst1
                DB2InstHome = "/db2_mnt/db2inst1"
                IndepthMonitor = 1
                DatabaseName = DWCNTRL
                NodeNumber = 0
                )

        Db2udb db2udb2 (
                DB2InstOwner = db2inst1
                DB2InstHome = "/db2_mnt/db2inst1"
                IndepthMonitor = 0
                NodeNumber = 1
                )
```

```
DiskGroup db2dg1 (
        DiskGroup = db2dg1
        )

IP db2ip1 (
        Device = en0
        Address = "166.98.9.188"
        NetMask = "255.255.220.0"
        )

Mount db2mnt1 (
        MountPoint = "/db2_mnt/db2inst1"
        BlockDevice = "/dev/vx/dsk/db2dg1/db2dg1home"
        FSType = vxfs
        MountOpt = rw
        )

NIC db2nic1 (
        Device = en0
        NetworkHosts = { "166.98.128.180" }
        )

Volume db2vol1 (
        Volume = db2dg1home
        DiskGroup = db2dg1
        )

db2ip1 requires db2nic1
db2mnt1 requires db2vol1
db2udb1 requires db2ip1
db2udb1 requires db2mnt1
db2vol1 requires db2dg1
db2udb2 requires db2ip1
db2udb2 requires db2mnt1
```

Sample MPP Configuration: DB2 UDB ESE Multi-Partition Instance

The following configuration file reflects DB2 UDB in an ESE multi-partition instance MPP environment. Four database partitions are shown. One partition is configured on each cluster node. Each database service group depends on the CVM service group, which manages the shared storage in the cluster.

```
include "types.cf"
include "CFSTypes.cf"
include "CVMTypes.cf"
include "Db2udbTypes.cf"

cluster db2_aix_mpp (
        UserNames = { admin = gpqIpkPmqLqqOyqKpn }
        Administrators = { admin }
        HacliUserLevel = COMMANDROOT
        CounterInterval = 5
        )

system sysA (
        )

system sysB (
        )

group cvm (
        SystemList = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
        AutoFailOver = 0
        Parallel = 1
        AutoStartList = { sysA, sysB, sysC, sysD }
        )

        CFSMount db2cfsmnt (
             MountPoint = "/db2_mnt/db2inst1"
             BlockDevice = "/dev/vx/dsk/cdb2dg1/cdb2dg1home"
             MountOpt = "cluster"
             NodeList = { sysA, sysB, sysC, sysD }
             )

        CFSfsckd vxfsckd (
          )
```

```
CVMCluster cvm_clus (
     CVMClustName = db2_aix_mpp
     CVMNodeId = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
     CVMTransport = gab
     CVMTimeout = 200
     )

CVMVolDg db2dg (
     CVMDiskGroup = cdb2dg1
     CVMActivation = sw
     )

CVMVxconfigd cvm_vxconfigd (
     Critical = 0
     CVMVxconfigdArgs = { syslog }
     )

cvm_clus requires cvm_vxconfigd
db2cfsmnt requires db2dg
db2cfsmnt requires vxfsckd
db2dg requires cvm_clus
vxfsckd requires cvm_clus


// resource dependency tree
//
//      group cvm
//      {
//      CFSMount db2cfsmnt
//         {
//         CVMVolDg db2dg
//         {
//         CVMCluster cvm_clus
//            {
//            CVMVxconfigd cvm_vxconfigd
//            }
//         }
//         CFSfsckd vxfsckd
//            {
//            CVMCluster cvm_clus
//               {
//               CVMVxconfigd cvm_vxconfigd
//               }
//            }
//         }
//      }
```

```
group db2mpp_grp0 (
        SystemList = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
        AutoStartList = { sysA }
        )

        Db2udb db2udb0 (
             DB2InstOwner = db2inst1
             DB2InstHome = "/db2_mnt/db2inst1"
             IndepthMonitor = 1
             DatabaseName = sample
             )

        IP Db2_IP0 (
             Device = en0
             Address = "11.192.10.32"
             NetMask = "255.255.244.0"
             )

        NIC mynic0 (
             Device = en0
             NetworkHosts = { "11.192.11.90" }
             )

        requires group cvm online local firm
        Db2_IP0 requires mynic0
        db2udb0 requires Db2_IP0


        // resource dependency tree
        //
        //      group db2mpp_grp0
        //      {
        //      Db2udb db2udb0
        //         {
        //         IP Db2_IP0
        //            {
        //            NIC mynic0
        //            }
        //         }
        //      }
```

```
group db2mpp_grp1 (
        SystemList = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
        AutoStartList = { sysB }
        )

        Db2udb db2udb1 (
             DB2InstOwner = db2inst1
             DB2InstHome = "/db2_mnt/db2inst1"
             IndepthMonitor = 1
             DatabaseName = test1
             NodeNumber = 1
             )

        IP Db2_IP1 (
             Device = en0
             Address = "11.192.10.33"
             NetMask = "255.255.244.0"
             )

        NIC mynic1 (
             Device = en0
             NetworkHosts = { "11.192.11.90" }
             )

        requires group cvm online local firm
        Db2_IP1 requires mynic1
        db2udb1 requires Db2_IP1


        // resource dependency tree
        //
        //      group db2mpp_grp1
        //      {
        //      Db2udb db2udb1
        //         {
        //         IP Db2_IP1
        //            {
        //            NIC mynic1
        //            }
        //         }
        //      }
```

```
group db2mpp_grp2 (
        SystemList = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
        AutoStartList = { sysC }
        )

        Db2udb db2udb2 (
             DB2InstOwner = db2inst1
             DB2InstHome = "/db2_mnt/db2inst1"
             IndepthMonitor = 1
             DatabaseName = test2
             NodeNumber = 2
             )

        IP Db2_IP2 (
             Device = en0
             Address = "11.192.10.33"
             NetMask = "255.255.244.0"
             )

        NIC mynic2 (
             Device = en0
             NetworkHosts = { "11.192.11.90" }
             )

        requires group cvm online local firm
        Db2_IP2 requires mynic2
        db2udb2 requires Db2_IP2


        // resource dependency tree
        //
        //      group db2mpp_grp2
        //      {
        //      Db2udb db2udb2
        //         {
        //         IP Db2_IP2
        //            {
        //            NIC mynic2
        //            }
        //         }
        //      }
```

```
group db2mpp_grp3 (
        SystemList = { sysA = 0, sysB = 1, sysC = 2, sysD = 3 }
        AutoStartList = { sysD }
        )

        Db2udb db2udb3 (
             DB2InstOwner = db2inst1
             DB2InstHome = "/db2_mnt/db2inst1"
             NodeNumber = 3
             )

        IP Db2_IP3 (
             Device = en0
             Address = "11.192.10.33"
             NetMask = "255.255.244.0"
             )

        NIC mynic3 (
             Device = en0
             NetworkHosts = { "11.192.11.90" }
             )

        requires group cvm online local firm
        Db2_IP3 requires mynic3
        db2udb3 requires Db2_IP3


        // resource dependency tree
        //
        //      group db2mpp_grp3
        //      {
        //      Db2udb db2udb3
        //         {
        //         IP Db2_IP3
        //            {
        //            NIC mynic3
        //            }
        //         }
        //      }
```

Index