Veritas™ High Availability Agent for DB2 Installation and Configuration Guide

AIX, Linux, Solaris

5.0

N18455C

Veritas High Availability Agent for DB2 Installation and Configuration Guide

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014
www.symantec.com

Third-party legal notices

Technical support

For technical assistance, visit http://support.veritas.com and select phone or email support. Use the Knowledge Base search feature to access resources such as TechNotes, product alerts, software downloads, hardware compatibility lists, and our customer email notification service.

Contents

Introduction

The Veritas High Availability agent, version 5.0, 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.

New features

This release features a new installation path for the agent for DB2. The path is /opt/VRTSagents/ha/. The change in the path is to better align it for consistency across the Symantec product line.

The attributes StartUpOpt and ShutDownOpt provide new start up and shut down options. Using the StartUpOpt attribute, you can start the instance or partition, activate database commands after processes start, or create customized start up sequences. Using the ShutDownOpt attribute, you can perform a normal stop or customize your shut down sequence.

See “Db2udb resource type attributes” on page 36.

In previous releases when you enabled in-depth monitoring (IndepthMonitor=1), it executed a default SQL query. The in-depth monitor now allows you to classify action for DB2 errors according to their severity. You can associate predefined actions with each error code with a monitoring script that you can customize. You can find a sample of in-depth monitoring script is in the /etc/VRTSagents/ha/conf/Db2udb/sample_db2udb directory. Install the custom script in the /opt/VRTSagents/ha/bin/Db2udb directory.

See “Handling DB2 error codes during in-depth monitoring” on page 11 and “Db2udb resource type attributes” on page 36.

You can enable the AgentDebug attribute to get more debugging information from the agent and the database.

See “Db2udb resource type attributes” on page 36.

Version numbers and operating systems

The Veritas High Availability agent, version 5.0 for DB2 UDB, supports DB2 Universal Database Enterprise Server Edition (ESE) versions 8.1, 8.2, and 9.1 for single and multi-partition instance. For ESE multi-partition instance, it supports both the Symmetric Multiprocessing (SMP) hardware configuration and the Massively Parallel Processing (MPP) hardware configuration.

The agent for DB2 ESE must run on:

- Solaris 2.8 or later
- AIX 5.2 or later
- RHEL 4.0
- SLES 9.0

The Veritas Cluster Server (VCS) version must be 5.0 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 agent for DB2 UDB

The agent for DB2 monitors DB2 database instances and databases 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 instances, the agent brings the DB2 UDB database instances online, monitors database processes, and shuts them down. For ESE multi-partition instances, the agent brings the DB2 UDB database partitions online, monitors the database processes at the partition level, and shuts down the database partitions.

The agent for DB2 operations

The agent for DB2 performs online, offline, monitor, clean, action, and info operations depending on:

- the version of DB2 UDB is ESE single-partition instance, or
- the ESE multi-partition instance and the ESE version

Online operation

For ESE version 8 and higher, 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

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.

Handling DB2 error codes during in-depth monitoring

The agent for DB2 comes with enhanced handling of DB2 errors during in-depth monitoring. The agent classifies DB2 errors according to their severity and associates predefined actions with each error code.

The agent includes a reference file called db2error.dat, which you can customize. The file lists the DB2 errors and the associated actions that you want the agent to take when it encounters an error.

The file stores information in the following format:

```
SQL_error_string:action_to_be_taken
```

For example:

```
SQL1034N: IGNORE
SQL1039N: WARN
SQL1234N: FAILOVER
```

The agent supports the following actions:

Action	Description
`IGNORE`	Ignores the error.

Action	Description
`UNKNOWN`	Marks the resource state as UNKNOWN and sends a notification if the Notifier resource is configured. See the *Veritas Cluster Server User's Guide* for more information about VCS notification. This action is typically associated with configuration errors.
`WARN`	Marks the resource state as ONLINE and sends a notification if the Notifier resource is configured. This action is typically associated with low-severity errors.
`FAILOVER` (Default)	Marks the resource state as OFFLINE. This faults the service group, which fails over to the next available system. This is the agent's default behavior. If the DB2 error code encountered does not exist in the db2error.dat file, then the agent assumes this default behavior.
`NOFAILOVER`	Freezes the service group temporarily and marks the resource state as OFFLINE. The agent also sends a notification if the Notifier resource is configured. This action is typically associated with errors that are not system-specific. For example, if a database was corrupted, failing it over to another node does not help.

ESE configurations, version 8.0 or higher

For all DB2 ESE configurations, version 8.0 or higher, for a single or multi-partition instance in SMP or MPP configuration, the agent executes the `db2gcf -s -i $DB2InstOwner -p $nodenum` command 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 starting from ESE version 8.

Set the IndepthMonitor attribute to 1 for in-depth monitoring. Before this release, IndepthMonitor performed a default SQL query to the database. In 5.0, this default query no longer exists.

The agent now looks for the custom_monitor_$db2instance_$nodenum file in the /opt/VRTSagents/ha/bin/Db2udb directory. It executes this customized in-depth monitor file if the file exists and is executable. You can find samples of custom monitor scripts in the /etc/VRTSAgents/ha/conf/sample_db2udb directory.

Note: If the DB2 Instance is running inside of a Solaris 10 non-global zone, then you must create the custom_monitor_$db2instance_$nodenum file inside the non-global zone. In other words, you must create the file after doing a "`zlogin zonename`" to login to the zone, and then change to the /etc/VRTSagents/ha/bin/Db2udb directory that is relative to the local zone.

If the custom monitor has any errors or problems, it checks the value of the WarnOnlyIfDBQueryFailed attribute of the Db2udb agent. If you have a db2error.dat file in the /opt/VRTSagents/ha/bin/Db2udb directory, the agent checks this file, and handles the error according to the error configuration. For error handling information:

See “Handling DB2 error codes during in-depth monitoring” on page 11.

If you set the WarnOnlyIfDBQueryFailed attribute to 1 (its default), and you have configured the Notifier resource, the agent sends a notification, and returns exit code 110.

If you set the WarnOnlyIfDBQueryFailed attribute to 0, it performs error handling in the db2error.dat file.

Clean operation

For ESE version 8 and higher, 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 operation

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

To retrieve database information

1 Make the configuration writable:

```
# haconf -makerw
```

2 Specify the periodic interval in seconds that the info agent function is invoked. The default value of 0 means info agent function 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 db2udb1 ResourceInfo
State Valid
Msg
PARTITION: 0
    PID    TTY  TIME CMD
  413924    -   0:00 db2sysc

    TS Fri Jan 14 18:11:52 2005
```

For more information about the `info` agent function, refer to the *Veritas Cluster Server User's Guide* and the *Veritas Cluster Server Agent Developer's Guide*.

Action operation

The agent for DB2 supports the Action operation, 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 these predefined actions:

Table 1-1 Predefined action tokens and descriptions

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 db2udb1 VRTS_GetInstanceName -sys systemName
VCS NOTICE V-16-13323 Resource (db2udb0): action
(VRTS_GetInstanceName) completed successfully. Output is:

db2inst1
```

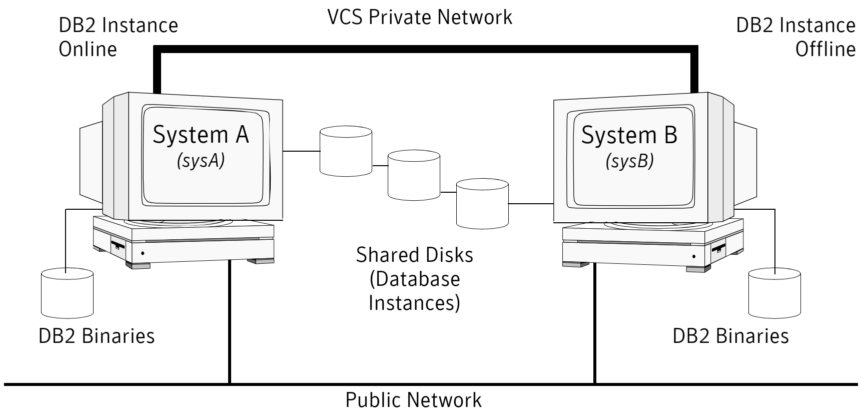
```
# hares -action db2udb0 VRTS_GetRunningServices -sys systemName

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

    PARTITION: 0

       PID TTY         TIME  CMD
       9800 ?          0:06  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

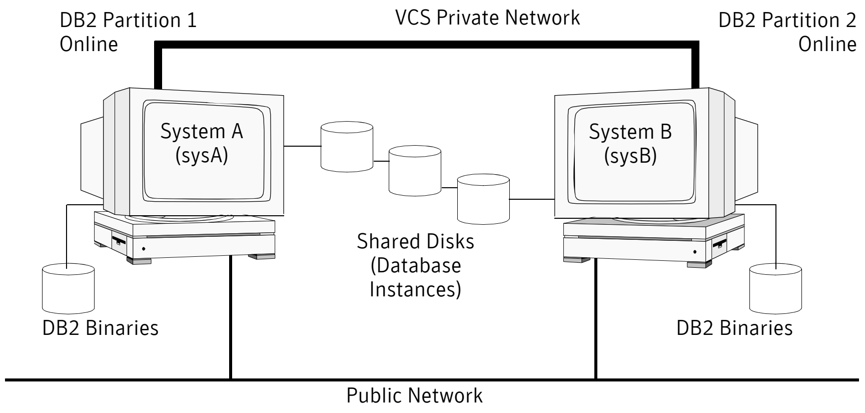
In the following examples, VCS is configured on a two-system cluster. DB2 UDB system binaries are installed identically on local file systems on System A and System B. The instance home directory, instance binaries, and the database reside on shared storage, available to either node. In the case of the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

DB2 Installation, non-MPP Configuration



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

DB2 Installation, MPP Configuration



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 5.0. Refer to the *Veritas Cluster Server Installation Guide* for instructions on running either the 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 23.
  - 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 Veritas agent for DB2, 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 installer or the installsfcfs utility.
- Before installing DB2 UDB, define DB2 UDB user and group accounts.
  See:
  - The relevant *DB2 Universal Database Quick Beginnings* guide.
  - "Defining DB2 user and group accounts" on page 23.

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, first create a disk group on the physically shared disk, and create a volume of sufficient size within the disk group.

To create a file system on AIX systems for non-MPP instances

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 /dev/vx/dsk/db2db_dg/db2db_vol \
  /db2_mnt/db2inst1
```

To create a file system on Linux systems for non-MPP instances

1 Create a disk group on the shared disk. List the disks using the `vxdisk list` command. In this case the group consists of one disk, sdc. For example:

```
# vxdg init db2db_dg /dev/sdc
```

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 -t vxfs /dev/vx/dsk/db2db_dg/db2db_vol
```

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

```
# mkdir /db2_mnt/db2inst1
# mount -t vxfs /dev/vx/dsk/db2db_dg/db2db_vol \
  /db2_mnt/db2inst1
```

5 Create the mount point directory and mount the file system:

```
# mkdir /db2_mnt/db2inst1
# mount -t vxfs /dev/vx/dsk/db2db_dg/db2db_vol \
  /db2_mnt/db2inst1
```

To create a file system on Solaris systems for non-MPP instances

1 Create a disk group on the shared disk. List the disks using the `vxdisk list` command. In this case the group consists of one disk, c4t0d0s2. For example:

```
# vxdg init db2db_dg c4t0d0s2
```

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 -F vxfs /dev/vx/rdsk/db2db_dg/db2db_vol
```

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

```
# mkdir /db2_mnt/db2inst1
# mount -F vxfs /dev/vx/dsk/db2db_dg/db2db_vol
    /db2_mnt/db2inst1
```

5 Create the mount point directory and mount the file system:

```
# mkdir /db2_mnt/db2inst1
# mount -F vxfs /dev/vx/dsk/db2db_dg/db2db_vol \
  /db2_mnt/db2inst1
```

Creating the shared cluster file system for the DB2 MPP instances

To create a shared file system, 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.

To create a shared file system on AIX for MPP instances

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:

```
cluster active - MASTER
```

Or

```
cluster active - SLAVE
```

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. In this example the disk is hdisk5:

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

4 Deport and import the disk group:

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

5 Use the `vxassist` command to create a seven GB volume:

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

6 Create the file system:

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

7 Create the mount point directory and mount the file system.

```
# mkdir /db2_mnt/db2inst1
# mount -V vxfs -o cluster /dev/vx/dsk/db2db_dg/db2db_vol \
  /db2_mnt/db2inst1
```

To create a shared file system on Linux for MPP instances

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:

```
cluster active - MASTER
```

Or

```
cluster active - SLAVE
```

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. In this example the disk is sdc:

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

4 Deport and import the disk group:

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

5 Use the `vxassist` command to create a seven GB volume:

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

6 Create the file system:

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

7 Create the mount point directory and mount the file system.

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

To create a shared file system on Solaris for MPP instances

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:

```
cluster active - MASTER
```

Or

```
cluster active - SLAVE
```

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. In this example the disk is c5t0d0s2:

```
# vxdg -s init db2db_dg c5t0d0s2
```

4 Deport and import the disk group:

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

5 Use the `vxassist` command to create a seven GB volume:

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

6 Create the file system:

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

7 Create the mount point directory and mount the file system.

```
mkdir /db2_mnt/db2inst1
mount -F vxfs -o cluster /dev/vx/dsk/db2db_dg/db2db_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.
- The DB2 instance owner's home directory, which is the mount point used by the DB2 instance created on shared storage, must exist on each node. If it does not, create the mount point directory on each node.
- 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 on each node in the cluster.

To create the group accounts on *each* node in the cluster for AIX systems

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
```

To create the group accounts on *each* node in the cluster for Linux and Solaris systems

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

   ```
   # groupadd -g 999 db2iadm1
   ```

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

   ```
   # groupadd -g 998 db2fadm1
   ```

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

   ```
   # groupadd -g 997 db2asgrp
   ```

Adding user accounts for AIX systems

Create the user accounts on each node in the cluster.

To create the user accounts on each node in the cluster for AIX systems

■ 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
```

■ 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
```

Adding user accounts for Linux and Solaris systems

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.

To create the user accounts on each node in the cluster for Linux and Solaris systems

■ 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:

```
# useradd -g db2iadm1 -u 1004 -d /db2_mnt/db2inst1 -m -s
/bin/ksh/ db2inst1
```

- 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.

```
# useradd -g db2fadm1 -u 1003 -d /home/db2fenc1 -m -s /bin/
ksh db2fenc1
# useradd -g db2asgrp -u 1002 -d /home/db2as -m -s /bin/ksh
db2as
```

Installing DB2 UDB in a VCS environment

For installing DB2 UDB version 8 and above ESE in a VCS environment, Symantec recommends that 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 in the /etc/system file on each node, 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:

```
# /opt/IBM/db2/V8.1/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, use the `more` command to check the file /etc/services.

- 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 two DB2 UDB instances: db2inst1 and db2inst2. Both instances have two partitions each. Each instance requires two ports plus one port per partition, hence four lines per instance.

```
# more /etc/services
DB2_db2inst1      60000/tcp
DB2_db2inst1_1    60001/tcp
DB2_db2inst1_2    60002/tcp
DB2_db2inst1_END  60003/tcp
DB2_db2inst2      60004/tcp
DB2_db2inst2_1    60005/tcp
DB2_db2inst2_2    60006/tcp
DB2_db2inst2_END  60007/tcp
```

Inspect the file and verify there are no duplicate port numbers.

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.

Configuring ssh on Suse

To configure ssh on Suse

1. Log on to the system from which you want to install VCS.
2. Generate a DSA key pair on this system by running the following command:
   ```
   # ssh-keygen -t dsa
   ```
3. Accept the default location: ~/.ssh/id_dsa
4. When prompted, enter a passphrase and confirm it.
5. Change the permissions of the .ssh directory, type:
   ```
   # chmod 755 ~/.ssh
   ```
6. The file ~/.ssh/id_dsa.pub contains a line beginning with ssh_dss and ending with the name of the system on which it was created. Copy this line to the /root/.ssh/authorized_keys2 file on all systems where VCS is to be installed.

   Note: If the local system is part of the cluster, make sure to edit the authorized_keys2 file on that system.

7. Run the following commands on the system from which the installation is taking place:
   ```
   # exec /usr/bin/ssh-agent $SHELL
   # ssh-add
   ```

   Note: This step is shell-specific and is valid for the duration the shell is alive.

8. When prompted, enter your DSA passphrase.

You are ready to install VCS on several systems by running the installvcs script on any one of them or on an independent machine outside the cluster.

To avoid running the ssh-agent on each shell, run the X-Window system and configure it so that you will not be prompted for the passphrase. Refer to the Red Hat documentation for more information.

Modifying the $DB2InstHome/sqllib/db2nodes.cfg file

DB2 uses the $DB2InstHome/sqllib/db2nodes.cfg file 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. DB2 automatically changes and updates the db2nodes.cfg file 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.

For example:

```
0 sysA 0
1 sysB 0
2 sysC 0
3 sysD 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 sysC 0
4 sysD 0
```

Confirming the setup of DB2 MPP and non-MPP installations

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

To check if a DB2 instance can start and stop

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
   ```

To check the rest of the DB2 configuration in the cluster

1. For each node in the VCS cluster, import the disk group and start all the volumes in the disk group.
2. Mount the file system for the volume containing the DB2 instance and database.
3. Perform the steps in the procedure above.
   See "To check if a DB2 instance can start and stop" on page 29.
4. Unmount and deport the disk group.
5. Repeat this procedure for each node in the cluster.

Installing the agent for DB2

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

Installing the DB2 UDB agent software

To install the agent on AIX systems

1 Log in as superuser.

2 Determine the device access name of the disc drive. For example, enter:

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

The output resembles:

```
cd0 Available 10-60-00-4,0 16 Bit SCSI Multimedia CD-ROM Drive
```

In this example, cd0 is the drive's device access name.

3 Insert the software disc containing the DB2 UDB agent software into the system's disc drive.

4 Mount the software disc using the device access name found in step 2:

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

5 Install the DB2 UDB agent software by entering:

```
# installp -ac -d /cdrom/VRTSvcsdb.rte.bff VRTSvcsdb.rte
```

6 Repeat this procedure on each system that will become part of the DB2 service group.

To install the agent on Linux systems

1 Log in as superuser.

2 Insert the software disc the system's drive. The disc automatically mounts.
If the disc does not automatically mount, enter:
```
# mount -o ro /dev/cdrom /mnt/cdrom
```

3 Go to the directory where rpm is present:
- On RHEL 4:
```
# cd /mnt/cdrom/rhel4_i686/cluster_server_agents/db2_agent/rpms
```
- On SLES 9:
```
# cd /mnt/cdrom/sles9_i586/cluster_server_agents/db2_agent/rpms
```

4 Install the DB2 UDB agent software:
```
# rpm -i VRTSvcsdb-5.0.00.10-GA_GENERIC.noarch.rpm
```

5 Verify the package is installed, enter: # rpm -q VRTSvcsdb
Where the output resembles:
```
VRTSvcsdb-5.0.00.10-GA_GENERIC
```

To install the agent on Solaris systems

1 Log in as superuser.

2 Create a temporary directory for installation:
```
# mkdir /tmp/install
```

3 Insert the disc into a drive connected to your system.
- If you are running Solaris volume-management software, the software automatically mounts the disc as /cdrom/cdrom0. Type the following command to go to the location: `# cd /cdrom/cdrom0`
- If you are not running Solaris volume-management software, you must mount the disc manually. For example:
```
    # mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom
```
  Where, in this example, /dev/dsk/c0t6d0s2 is the default for the CD drive.
  Once the disc is mounted, type the following commands to go to the location: `# cd /cdrom`

4 Copy the compressed package files from the software disc to the temporary directory:
```
# cp -r db2_agent/pkgs/* /tmp/install
```

5 Go to the temporary directory and unzip the compressed package file:

Note: If your system does not have the gunzip utility, copy it from the disc:

```
# cp /cdrom_path/gnu/gunzip /tmp/install
```

```
# cd /tmp/install
# gunzip VRTS*.gz
```

6 Extract the compressed file from the tar file:

```
# tar -xvf VRTSvcsdb.tar
```

7 Install the package:

```
# pkgadd -d . VRTSvcsdb
```

To install the Japanese language pack on Solaris systems

1 After installing the agent, insert the language disc into the system's drive. Type the command:

```
# cd /cdrom/cdrom0
```

2 Copy the compressed package files from the software disc to the temporary directory:

```
# cp -r ja/db2_agent/pkgs/* /tmp/install
```

3 Go to the temporary directory and unzip the compressed package file:

```
# cd /tmp/install
# gunzip VRTSjacsb.tar.gz
```

4 Extract the compressed file from the tar file:

```
# tar -xvf VRTSjacsb.tar
```

5 Install the Japanese package:

```
# pkgadd -d . VRTSjacsb
```

Configuring the agent for DB2

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 43.
- 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. This method requires that VCS be stopped and restarted before the configuration takes effect.
  See "Configuring the DB2 UDB agent by editing the main.cf file" on page 49.

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 Appendix A, "Sample configuration files".

Db2udb resource type attributes

The DB2 resource has several required and optional attributes.

Table 4-1 Required attributes for the agent for DB2

Required attributes	Description
DB2InstHome	Path to DB2 UDB instance home directory that contains configuration files for the DB2 instance. Type and dimension: string-scalar
DB2InstOwner	User ID of Instance Owner that starts a DB2 UDB instance. Each instance requires a unique user ID. Type and dimension: string-scalar **Caution:** Incorrect changes to this attribute can result in DB2 entering an inconsistent state.

Table 4-2 Optional attributes for the agent for DB2

Optional attributes	Description
DatabaseName	Name of the database for in-depth monitoring; required if in-depth monitor is enabled (IndepthMonitor = 1). **Note:** In an MPP environment, if this attribute changes all partitions reflect the change. Type and dimension: string-scalar
NodeNumber	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. Type and dimension: integer-scalar

Table 4-2 Optional attributes for the agent for DB2

Optional attributes	Description
StartUpOpt	Provides start up options. The allowed values are: START, ACTIVATEDB, or CUSTOM. ■ START (default) Starts the DB2 instance or partition. ■ ACTIVATEDB Performs activate database command after db2 processes start. ■ CUSTOM The agent leaves all the online operation completely to the user when the StartUpOpt attribute is set to CUSTOM. It looks for a file named start_custom_$db2instance_$nodenum in the /opt/VRTSagents/ha/bin/Db2udb directory. If this file exists and is executable, it executes this customized online file instead. Example: To customize the online operation for partition/ nodenum 1 for the db2 instance named db2inst1, the agent for DB2 runs this customized file start_custom_db2inst1_1 under the /opt/VRTSagents/ha/bin/Db2udb directory. Type and dimension: string-scalar
ShutDownOpt	The allowed values for this attribute are STOP and CUSTOM. ■ STOP Shuts the Db2 instance or partition down in the usual way. ■ CUSTOM Leaves all the offline operation completely to the user when the ShutDownOpt is set to CUSTOM. It looks for a file named stop_custom_$db2instance_$nodenum under the /opt/VRTSagents/ha/bin/Db2udb directory. If this file exists and is executable, it executes this customized offline file instead. Example: If you want to customize the offline operation for partition/nodenum 0 for the db2 instance named db2inst1, then the agent for DB2 runs this customized file stop_custom_db2inst1_0 under the /opt/VRTSagents/ha/bin/Db2udb directory. Type and dimension: string-scalar

Table 4-2 Optional attributes for the agent for DB2

Optional attributes	Description
IndepthMonitor	Set the value of the IndepthMonitor attribute to 1 to enable in-depth monitoring. Before this release, IndepthMonitor performed a default SQL query to the database. In 5.0, this default query no longer exists. The agent now looks for the custom_monitor_$db2instance_$nodenum file in the /opt/VRTSagents/ha/bin/Db2udb directory. It executes this customized indepth monitor file if the file exists and is executable. You can find samples of custom monitor scripts in the sample_db2udb directory. Type and dimension: string-integer
Encoding	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,” then “eucJP” is the Solaris value for Encoding. Refer to DB2 UDB and Solaris documentation for respective encoding values. The default is “”. Type and dimension: string-scalar
AgentDebug	When set to 1, causes the agent to log additional debug messages. Type and dimension: boolean-scalar
WarnOnlyIfDBQueryFailed	This attribute either logs SQL errors, or checks the errors for special handling. Set the value of the WarnOnlyIfDBQueryFailed attribute to 1 to enable it. When enabled, it ignores all SQL errors and logs a warning message in the agent log once a day. Set the value of the WarnOnlyIfDBQueryFailed attribute to 0 to disable it. When disabled, it checks if an error code has special handling in the db2error.dat file. If the error code does not exist in the db2error.dat file, then it returns OFFLINE for monitor. Otherwise, it follows the action of that particular error code in the db2error.dat file. Type and dimension: boolean-scalar
ContainerName	Name of the Solaris zone (Solaris 10 only) Type and dimension: string-scalar

Table 4-3 Internal attributes for the agent for DB2

Required attributes	Description
AgentDirectory	Specifies the location of other files and scripts related to the agent. Do not use. For internal use only.

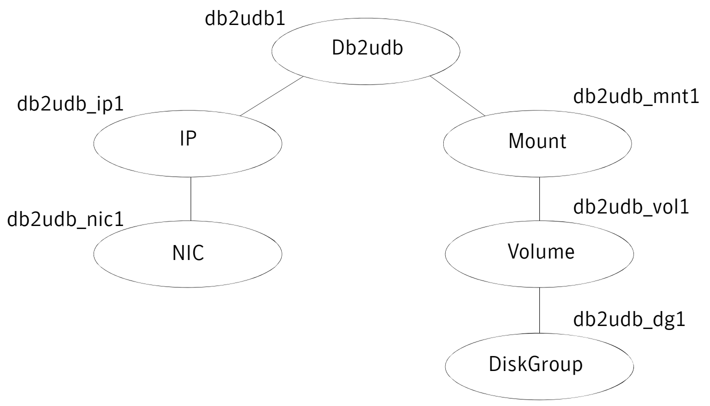
DB2 UDB type definition file: Db2udbTypes.cf for Solaris

```
type Db2udb (
    static str ContainerType = Zone
    static str AgentDirectory = "/opt/VRTSagents/ha/bin/Db2udb"
    static str AgentFile = "/opt/VRTSagents/ha/bin/Db2udb/
    Db2udbAgent"
    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 = 3
    static int ToleranceLimit = 1
    static str ArgList[] = { DB2InstOwner, DB2InstHome,
    IndepthMonitor, DatabaseName, NodeNumber, StartUpOpt,
    ShutDownOpt, AgentDebug, Encoding, WarnOnlyIfDBQueryFailed,
    LastWarningDay, ContainerName }
    str DB2InstOwner
    str DB2InstHome
    int IndepthMonitor
    str DatabaseName
    int NodeNumber
    str StartUpOpt = START
    str ShutDownOpt = STOP
    boolean AgentDebug = 0
    str Encoding
    boolean WarnOnlyIfDBQueryFailed = 1
    temp str LastWarningDay
    str ContainerName
)
```

DB2 UDB service group for non-MPP configuration

Figure 4-1 illustrates the dependencies among the resources configured for a non-MPP DB2 UDB instance resource group.

Figure 4-1 Dependency tree for a Db2udb instance



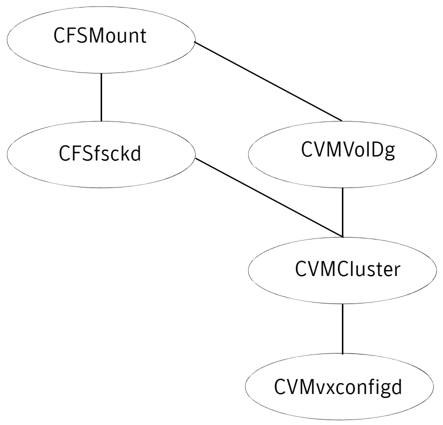
This configuration shows a service resource group for an instance of DB2 UDB. The db2udb1 resource (the database) requires the IP resource and the Mount resource. The service group IP address for the DB2 UDB server is configured using the IP resource (db2udb_ip1) and the NIC resource (db2udb_nic1). The 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.

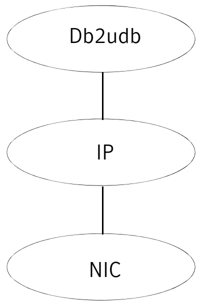
The first service group is a parallel CVM service group. There is one CVM/ Infrastructure group per cluster node. This service group has the CVM resource and the necessary resources for support of CFS. This group also contains all common components needed by DB2, such as the instance's home directory,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

Figure 4-2 Parallel CVM service group



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.

Figure 4-3 DB2 Failover Service Group



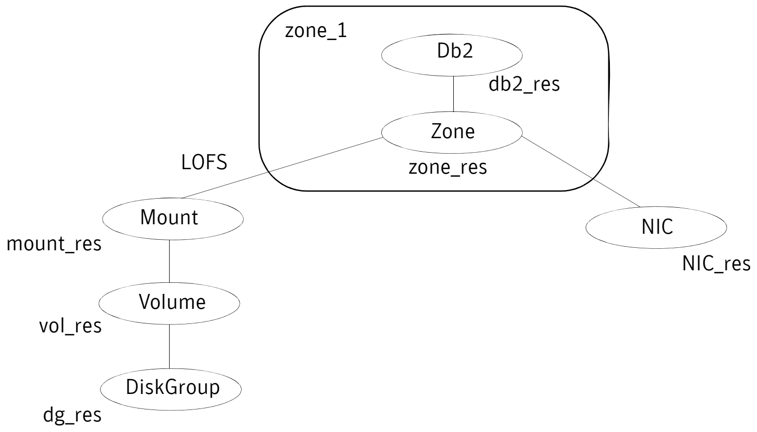
DB2 UDB instance configured in Solaris zones

The following examples describe a typical service group configured to monitor the state of a DB2 instance configured in a Solaris zone.

Zone root on local disk

If the root file system of a zone is on the local disk of each node, the file system mounts when you boot the system. Hence, the service group does not need to have separate DiskGroup and Volume resources for the zone.

Figure 4-4 DB2 UDB instance configured in a Solaris zone



The shared disk groups and volumes in the cluster are configured as resources of type DiskGroup and Volume respectively. The volumes are mounted using the Mount agent. The Solaris zone is monitored through a zone resource, which is dependent on the Mount and NIC resources. The DB2 server can be started after each of these resources is brought online.

The DB2 instance's home directory is mounted in the global zone. In order to make this file system available to the non-global zone, you must execute the following command on the global zone.

The lines in the following example specify that you mount /zones/db2data in the global zone as /db2inst1 in the non-global zone named zone1. The file system type to use is LOFS. The /db2inst1 directory in this example is the home directory for the DB2 instance.

```
# zonecfg -z zone1
zonecfg:zone1> add fs
zonecfg:zone1:fs> set dir=/db2inst1
zonecfg:zone1:fs> set type=lofs
```

```
zonecfg:zone1:fs> set special=/zones/db2data
zonecfg:zone1:fs> end
```

For more information on setting up VCS to work with Solaris 10 local zones, see the appendix in the *VCS User's Guide* on configuring VCS in non-global zones.

Configuring the DB2 UDB agent using cluster manager

Templates for the DB2 UDB resource groups were automatically installed when you installed the agent for DB2. Using the VCS Cluster Manager (Java Console), you can use the template (/etc/VRTSagents/ha/Templates/Db2udbGroup.tf) 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.

Importing the Db2udbTypes.cf file

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

To import the Db2udbTypes.cf file

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/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf
   ```
5. Click **Import** and wait for the file to import.
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:
     See "Adding service group for DB2 UDB MPP using Cluster Manager" on page 44.
   - If you are using the DB2 UDB non-MPP configuration:
     See "Adding a service group for DB2 UDB non-MPP using Cluster Manager" on page 46.

Adding service group for DB2 UDB MPP using Cluster Manager

If you have imported the Db2udbTypes.cf file, you can use the template (/etc/VRTSagents/ha/Templates/Db2udbGroup.tf) to configure a service group.

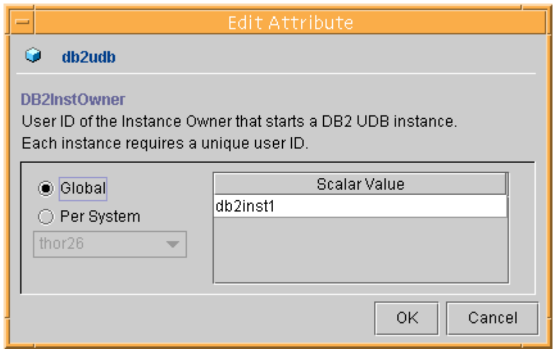
See "Importing the Db2udbTypes.cf file" on page 43.

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

1. In the Cluster Explorer window, click the **Add Service Group** icon on the toolbar.
2. In the Add Service Group window, 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**.
3. In the Available Systems box, double-click the systems that you want in your configuration.
4. In the window showing the systems added to the configuration, click the checkbox for system where you want to automatically start the service group.
5. Click the **Failover** radio button to specify the Service Group Type.
6. Click the Templates button.
7. 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.
8. 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.
9. In the left pane, double-click the db2mpp_grp1 service group. The types of resources that you can configure for the group appear: Db2udb, IP, and NIC.
10. Double-click the Db2udb resource type. Select the resource, db2udb, below the Db2udb type and click the **Properties** tab.
11. 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

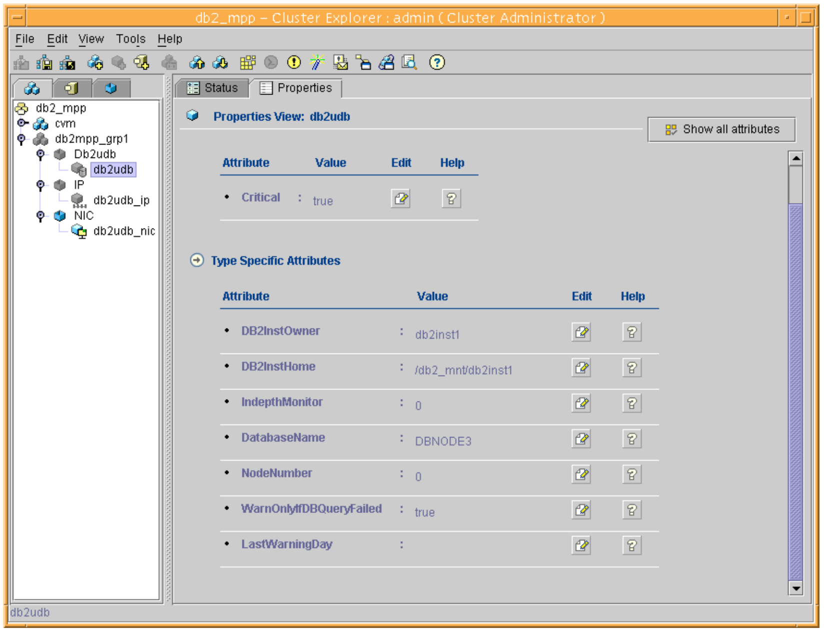
db2udb

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

(●) Global
( ) Per System
thor26

Scalar Value
db2inst1

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



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			

12 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.

13 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.”

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

15 Click the **Save Configuration** icon.

16 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.

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

18 In the window, select the service group and the system that you want to bring online. Click **OK**.

Adding a service group for DB2 UDB non-MPP using Cluster Manager

If you have imported the Db2udbTypes.cf file, you can use the Db2udb_Group template to configure a service group.

See “Importing the Db2udbTypes.cf file” on page 43.

To configure a service group if you have imported the Db2udbTypes.cf file

1 In the Cluster Explorer window, answer **No** when prompted to use the configuration wizard. Note that 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, 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**.

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

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

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

7 Click the **Templates** button.

8 In the Select Templates window, select db2udb_grp from the list shown in the Templates box. The Dependency graph information and the Types

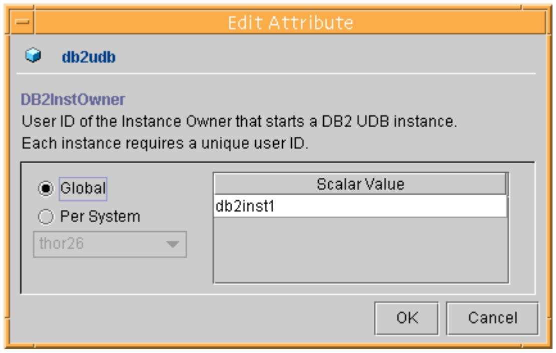
information should change to reflect the template choice. Click **OK**. In the Add Service Group window, the name of the template is now shown as selected.

9 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 under the cluster name. On the Status tab, the group is shown Offline on each system.

10 In the left pane, double-click the **db2_group1** service group. The types of resources that you can configure for the group appear: Db2udb, DiskGroup, IP, Mount, NIC, and Volume.

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

12 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.

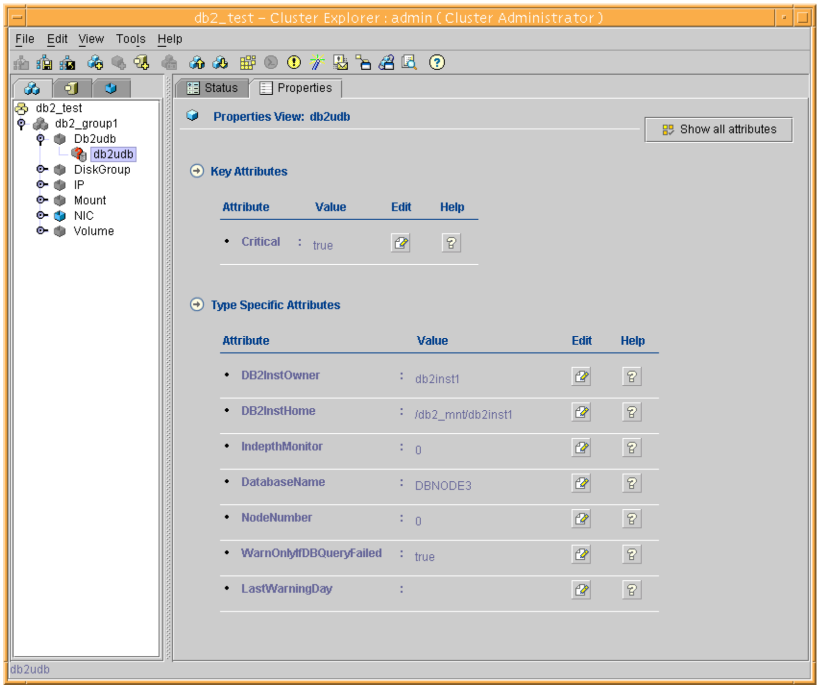
[✓ Global] Per System

thor26

Scalar Value
db2inst1

Note: DB2InstOwner and DB2InstHome are required attributes. You must edit these attributes.

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



Attribute	Value	Edit	Help
Critical	true		

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

13 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.

For a list of the required attributes, and their descriptions, see the *VCS Bundled Agents Reference Guide*.

14 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.

15 Click the **Save and Close Configuration** icon.

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

17 In the window, select the service group and the system that you want to bring online. Click the system where you want to bring it online. Click **OK**. Click **Yes** at the confirmation question.

Configuring the DB2 UDB agent by editing the main.cf file

The VCS agent for DB2 comes with three sample VCS configuration files installed in the /etc/VRTSagents/ha/conf/Db2udb/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.

To prepare to edit 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 agent for DB2.

Configuring the agent to use the DB2 UDB MPP configuration

Edit the main.cf file. Use /etc/VRTSagents/ha/conf/Db2udb/sample_db2udb/main.cf.MPP for reference. Notice that CVM service group is present in the configuration file.

To configure the agent to use the DB2 UDB MPP configuration

1 Add the fully qualified path to the Db2udbTypes.cf file.

```
include "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"
```

2 Create service groups for the DB2 UDB resources. Refer to the sample configuration file. The example shows four DB2 MPP service groups and a CVM service group.
See “MPP main.cf configuration: DB2 UDB ESE multi-partition instance” on page 84.

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.
See:
   - "Db2udb resource type attributes" on page 36.
   - Sample configuration files.
   - The *VCS 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 "MPP main.cf configuration: DB2 UDB ESE multi-partition instance" on page 84, for the group `db2mpp_grp0`, enter:
```
db2udb0 requires Db2_IP0
Db2_IP0 requires Db2_NIC0
```

6 Save and close the file.

Configuring the agent to use the DB2 UDB, non-MPP configurations

Edit the main.cf file. Use /etc/VRTSagents/ha/conf/Db2udb/sample_db2udb/main.cf.EE or /etc/VRTSagents/ha/conf/Db2udb/sample_db2/main.cf.EEE for reference.

To configure the agent to use the DB2 UDB, non-MPP configurations

1 Add the fully qualified path to the Db2udbTypes.cf file.
```
include "/etc/VRTSagents/ha/conf/Db2udb/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, "Non-MPP main.cf configuration: DB2 UDB ESE single-partition instance" on page 69 which shows two groups, "db2udb_grp1," and "db2udb_grp2."
If you are using DB2 UDB ESE multi-partition instance, refer to the example, "Non-MPP main.cf configuration: DB2 UDB ESE multi-partition instance" on page 62, 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 36 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 "Non-MPP main.cf configuration: DB2 UDB ESE single-partition instance" on page 69, for the group db2udb_grp1, 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, 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.

To check the configuration

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

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

2 Start the VCS engine on System A:

```
# hastart
```

3 Type the hastatus command:

```
# hastatus
```

4 When "LOCAL_BUILD" is listed in the message column, start VCS on System B:

```
# hastart
```

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

   ```
   # hagrp -display
   ```

6. 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
   ```

7. 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
   ```

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

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

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

   ```
   # hagrp -display
   ```

10. 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 Veritas agent for DB2, 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. Symantec recommends 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.

For locales other than English, you need to add the following lines to the $INSTHOME/sqllib/*userprofile* file. The following example adds Japanese language support:

```
LANG=ja
export LANG
```

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
   ```

Note: You need to have custom monitoring scripts.
See "IndepthMonitor" on page 38.

Disabling in-depth monitoring

You can dynamically disable in-depth monitoring.

To dynamically disable in-depth monitoring

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, removing, and upgrading the agent

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.

To disable the agent

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

Before you remove the agent, you must disable it. Perform the following instructions to remove the agent.

To remove the agent from AIX systems

- On each system that has the agent, type:

  ```
  # installp -u VRTSvcsdb.rte
  ```

To remove the agent from Linux systems

- On each system that has the agent, type:

  ```
  # rpm -e VRTSvcsdb
  ```

To remove the agent from Solaris systems

- On each system that has the agent, type:

  ```
  # pkgrm VRTSvcsdb
  ```

Upgrading the agent

You can only upgrade the HA agent for DB2 manually. The installvcs program does not automatically upgrade the VRTSvcsdb package.

To upgrade without saving previous configuration information is to disable the agent, remove it, and re-install it.

The steps to upgrade and re-use previous configuration information follow.

To upgrade from DB2 agent 4.0 or 4.1

1. Disable and remove the agent.
   - See "Disabling the agent" on page 55.
   - See "Removing the agent" on page 56.
2. From the disc that has the HA agent for DB2, add the new package.
   See "Installing the DB2 UDB agent software" on page 31.
3. Copy the new "Db2udbTypes.cf" from the /etc/VRTSagents/ha/conf directory to the /etc/VRTSagents/ha/conf/config directory.
4. Update the location of the Db2udbTypes.cf file in your main.cf file's include statement. For example, change this statement:

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

   To read:

   ```
   include "/etc/VRTSagents/ha/conf/config/DB2udbTypes.cf"
   ```

5 To continue to use in-depth monitoring, use the custom monitoring sample script.
See “Handling DB2 error codes during in-depth monitoring” on page 11.

Sample configuration files

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

AIX sample configuration files

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/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/vg00"
        FSType = jfs
        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
        //                    }
        //              }
        //          }
```

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/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
```

MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/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
    //                   }
    //               }
    //          }
```

Linux sample configuration files

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"

cluster vcs (
    CounterInterval = 5
    )

system vcstc1 (
    CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
    ActionTimeLimit = 0, Action = NONE, NotifyThreshold = 0,
    NotifyTimeLimit = 0 }
    )

system vcstc2 (
    CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
    ActionTimeLimit = 0, Action = NONE, NotifyThreshold = 0,
    NotifyTimeLimit = 0 }
    )

group db2udb_grp1 (
    SystemList = { vcstc1= 0, vcstc2 = 1 }
    AutoStartList = { vcstc1 }
    )

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

    DiskGroup db2udb_dg1 (
        DiskGroup = db2_dg1
        )

    IP db2udb_ip1 (
        Device = eth0
        Address = "166.98.9.163"
        NetMask = "255.255.252.0"
        )

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

    NIC db2udb_nic1 (
        Device = eth0
        )

    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_grp3 (
    SystemList = { vcstc1 = 0, vcstc2 = 1 }
    AutoStartList = { vcstc1 }
    )

    Db2udb db2udb3 (
        DB2InstOwner = db2inst3
        DB2InstHome = "/db2inst3"
        IndepthMonitor = 1
        DatabaseName = MYDB
        )

    DiskGroup db2udb_dg3 (
        DiskGroup = db2_dg3
        )

    IP db2udb_ip3 (
        Device = eth0
        Address = "192.2.40.21"
        NetMask = "255.255.252.0"
        )

    Mount db2udb_mnt3 (
        MountPoint = "/db2inst3"
        BlockDevice = "/dev/vx/dsk/db2_dg3/inst3_vol"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-n"
        )

    NIC db2udb_nic3 (
        Device = eth0
        )

    Volume db2udb_vol3 (
        Volume = inst3_vol
        DiskGroup = db2_dg3
        )

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

    // resource dependency tree
    //
    //          group db2udb_grp3
    //          {
    //          Db2udb db2udb3
    //                {
    //                IP db2udb_ip3
    //                      {
    //                      NIC db2udb_nic3
    //                      }
    //                Mount db2udb_mnt3
    //                      {
    //                      Volume db2udb_vol3
    //                            {
    //                            DiskGroup db2udb_dg3
    //                            }
    //                      }
    //                }
    //          }
```

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"

cluster vcs (
    CounterInterval = 5
    )

system vcstc1 (
    CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
    ActionTimeLimit = 0, Action = NONE, NotifyThreshold = 0,
    NotifyTimeLimit = 0 }
    )

system vcstc2 (
    CPUUsageMonitoring = { Enabled = 0, ActionThreshold = 0,
    ActionTimeLimit = 0, Action = NONE, NotifyThreshold = 0,
    NotifyTimeLimit = 0 }
    )

group db2udb_grp1 (
    SystemList = { vcstc1= 0, vcstc2 = 1 }
    AutoStartList = { vcstc1 }
    )

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

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

```
DiskGroup db2udb_dg1 (
    DiskGroup = db2_dg1
    )

IP db2udb_ip1 (
    Device = eth0
    Address = "166.98.9.163"
    NetMask = "255.255.252.0"
    )

IP db2udb_ip2 (
    Device = eth0
    Address = "166.98.9.165"
    NetMask = "255.255.252.0"
    )

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

NIC db2udb_nic1 (
    Device = eth0
    )

Volume db2udb_vol1 (
    Volume = inst1_vol
    DiskGroup = db2_dg1
    )

db2udb1 requires db2udb_ip1
db2udb1 requires db2udb_mnt1
db2udb2 requires db2udb_ip2
db2udb2 requires db2udb_mnt1
db2udb_ip1 requires db2udb_nic1
db2udb_ip2 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
//                              }
//                        }
//                  }
//            Db2udb db2udb2
//                  {
//                  IP db2udb_ip2
//                        {
//                        NIC db2udb_nic1
//                        }
//                  Mount db2udb_mnt1
//                        {
//                        Volume db2udb_vol1
//                              {
//                              DiskGroup db2udb_dg1
//                              }
//                        }
//                  }
//            }
```

MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"

cluster db2_mpp (
    CounterInterval = 5
    )

system vcstc1 (
    )

system vcstc2 (
    )

system vcstc3 (
    )
```

```
system vcstc4 (
    )

group cvm (
    SystemList = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3 }
    AutoFailOver = 0
    Parallel = 1
    AutoStartList = { vcstc1, vcstc2, vcstc3, vcstc4 }
    )

    CFSMount db2cfsmnt (
        MountPoint = "/db2_mnt/db2inst1"
        BlockDevice = "/dev/vx/dsk/cdb2dg1/cdb2dg1home"
        )

    CFSfsckd vxfsckd (
        )

    CVMCluster cvm_clus (
        CVMClustName = db2_mpp
        CVMNodeId = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3
        }
        CVMTransport = gab
        CVMTimeout = 200
        )

    CVMVolDg db2dg (
        CVMDiskGroup = cdb2dg1
        CVMVolume = { cdb2dg1home }
        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 = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3 }
    AutoStartList = { vcstc1 }
    AutoStart = 1
    )

    Db2udb db2udb0 (
        DB2InstOwner = db2inst1
        DB2InstHome = "/db2_mnt/db2inst1"
        IndepthMonitor = 1
        DatabaseName = SAMPLE
        NodeNumber = 0
        )

    IP Db2_IP0 (
        Device = eth0
        Address = "10.118.2.144"
        NetMask = "255.255.248.0"
        )

    NIC Db2_NIC0 (
        Device = eth0
        )

    requires group cvm online local firm
    Db2_IP0 requires Db2_NIC0
    db2udb0 requires Db2_IP0

    // resource dependency tree
    //
    //          group db2mpp_grp0
    //          {
    //          Db2udb db2udb0
    //                {
    //                IP Db2_IP0
    //                      {
    //                      NIC Db2_NIC0
    //                      }
    //                  }
    //          }

group db2mpp_grp1 (
    SystemList = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3 }
    AutoStartList = { vcstc2 }
    AutoStart = 1
    )
    Db2udb db2udb1 (
        DB2InstOwner = db2inst1
        DB2InstHome = "/db2_mnt/db2inst1"
        IndepthMonitor = 1
        DatabaseName = TEST1
        NodeNumber = 1
        )

    IP Db2_IP1 (
        Device = eth0
        Address = "10.118.2.145"
        NetMask = "255.255.248.0"
        )

    NIC Db2_NIC1 (
        Device = eth0
        )

    requires group cvm online local firm
    Db2_IP1 requires Db2_NIC1
    db2udb1 requires Db2_IP1

    // resource dependency tree
    //
    //          group db2mpp_grp1
    //          {
    //          Db2udb db2udb1
    //                  {
    //                  IP Db2_IP1
    //                          {
    //                          NIC Db2_NIC1
    //                          }
    //                  }
    //          }

group db2mpp_grp2 (
    SystemList = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3 }
    AutoStartList = { vcstc3 }
    AutoStart = 1
    )

    Db2udb db2udb2 (
        DB2InstOwner = db2inst1
        DB2InstHome = "/db2_mnt/db2inst1"
```

```
            IndepthMonitor = 1
            DatabaseName = TEST2
            NodeNumber = 2
            )

        IP Db2_IP2 (
            Device = eth0
            Address = "10.118.2.146"
            NetMask = "255.255.248.0"
            )

        NIC Db2_NIC2 (
            Device = eth0
            )

        requires group cvm online local firm
        Db2_IP2 requires Db2_NIC2
        db2udb2 requires Db2_IP2


        // resource dependency tree
        //
        //          group db2mpp_grp2
        //          {
        //          Db2udb db2udb2
        //                {
        //                IP Db2_IP2
        //                      {
        //                      NIC Db2_NIC2
        //                      }
        //                }
        //          }

    group db2mpp_grp3 (
        SystemList = { vcstc1 = 0, vcstc2 = 1, vcstc3 = 2, vcstc4 = 3 }
        AutoStartList = { vcstc4 }
        AutoStart = 1
        )

        Db2udb db2udb3 (
            DB2InstOwner = db2inst1
            DB2InstHome = "/db2_mnt/db2inst1"
            NodeNumber = 3
            )
        IP Db2_IP3 (
            Device = eth0
            Address = "10.118.2.147"
            NetMask = "255.255.248.0"
            )
    NIC Db2_NIC3 (
        Device = eth0
        )

    requires group cvm online local firm
    Db2_IP3 requires Db2_NIC3
    db2udb3 requires Db2_IP3


    // resource dependency tree
    //
    //          group db2mpp_grp3
    //          {
    //          Db2udb db2udb3
    //                {
    //                IP Db2_IP3
    //                      {
    //                      NIC Db2_NIC3
    //                      }
    //                }
    //          }
```

Solaris sample configuration files

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/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
        StartUpOpt = ACTIVATEDB
        )

    DiskGroup db2udb_dg1 (
        DiskGroup = db2_dg1
        )

    IP db2udb_ip1 (
        Device = hme0
        Address = "166.98.9.163"
        )
    Mount db2udb_mnt1 (
        MountPoint = "/db2inst1"
        BlockDevice = "/dev/vx/dsk/db2_dg1/inst1_vol"
        FSType = vxfs
        MountOpt = rw
        FsckOpt = "-y"
        )

    NIC db2udb_nic1 (
        Device = hme0
        NetworkType = ether
        )

    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
        )

    DiskGroup db2udb_dg2 (
        DiskGroup = db2_dg2
        )

    IP db2udb_ip2 (
        Device = hme0
        Address = "192.2.40.21"
        )

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

    NIC db2udb_nic2 (
        Device = hme0
        NetworkType = ether
        )
    Volume db2udb_vol2 (
        Volume = inst2_vol
        DiskGroup = db2_dg2
        )

    db2udb2 requires db2udb_ip2
    db2udb2 requires db2udb_mnt2
    db2udb_ip2 requires db2udb_nic2
    db2udb_mnt2 requires db2udb_vol2
    db2udb_vol2 requires db2udb_dg2

    // resource dependency tree
    //
    //          group db2udb_grp2
    //          {
    //          Db2udb db2udb2
    //              {
    //              IP db2udb_ip2
    //                  {
    //                  NIC db2udb_nic2
    //                  }
    //              Mount db2udb_mnt2
    //                  {
    //                  Volume db2udb_vol2
    //                      {
    //                      DiskGroup db2udb_dg2
    //                      }
    //                  }
    //              }
    //          }
```

Non-MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/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 = hme0
        Address = "192.2.40.21"
        )

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

    NIC db2nic1 (
        Device = hme0
        NetworkType = ether
        )

    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
```

MPP main.cf 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 "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"

cluster db2_mpp (
    CounterInterval = 5
    )

system sysA (
    )

system sysB (
    )

system sysC (
    )

system sysD (
    )

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"
        Primary = sysD
        )
    CFSfsckd vxfsckd (
        )

    CVMCluster cvm_clus (
        Critical = 0
        CVMClustName = db2_mpp
        CVMNodeId = { sysA = 0, sysB = 1, sysC = 2,
           sysD = 3 }
        CVMTransport = gab
        CVMTimeout = 200
        )

    CVMVolDg db2dg (
        CVMDiskGroup = cdb2dg1
        CVMVolume = { cdb2dg1home }
        CVMActivation = sw
        )

    db2cfsmnt requires db2dg
    db2cfsmnt requires vxfsckd
    db2dg requires cvm_clus
    vxfsckd requires qlogckd

    // resource dependency tree
    //
    //          group cvm
    //          {
    //          CFSMount db2cfsmnt
    //              {
    //              CVMVolDg db2dg
    //                  {
    //                  CVMCluster cvm_clus
    //                  }
    //              CFSfsckd vxfsckd
    //                  {
    //                  CFSQlogckd qlogckd
    //                  }
    //              }
    //          }

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 = hme0
        Address = "10.118.2.144"
        NetMask = "255.255.248.0"
        )

    NIC Db2_NIC0 (
        Device = hme0
        NetworkHosts = { "10.118.11.90" }
        )

    requires group cvm online local firm
    Db2_IP0 requires Db2_NIC0
    db2udb0 requires Db2_IP0


    // resource dependency tree
    //
    //          group db2mpp_grp0
    //          {
    //          Db2udb db2udb0
    //              {
    //              IP Db2_IP0
    //                  {
    //                  NIC Db2_NIC0
    //                  }
    //              }
    //          }


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 = hme0
        Address = "10.118.2.145"
        NetMask = "255.255.248.0"
        )
    NIC Db2_NIC1 (
        Device = hme0
        NetworkHosts = { "10.118.11.90" }
        )

    requires group cvm online local firm
    Db2_IP1 requires Db2_NIC1
    db2udb1 requires Db2_IP1


    // resource dependency tree
    //
    //         group db2mpp_grp1
    //         {
    //         Db2udb db2udb1
    //             {
    //             IP Db2_IP1
    //                 {
    //                 NIC Db2_NIC1
    //                 }
    //             }
    //         }


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 = hme0
        Address = "10.118.2.146"
        NetMask = "255.255.248.0"
        )

    NIC Db2_NIC2 (
        Device = hme0
        NetworkHosts = { "10.118.11.90" }
        )

    requires group cvm online local firm
    Db2_IP2 requires Db2_NIC2
    db2udb2 requires Db2_IP2
    // resource dependency tree
    //
    //          group db2mpp_grp2
    //          {
    //          Db2udb db2udb2
    //              {
    //              IP Db2_IP2
    //                  {
    //                  NIC Db2_NIC2
    //                  }
    //              }
    //          }

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 = hme0
        Address = "10.118.2.147"
        NetMask = "255.255.248.0"
        )

    NIC Db2_NIC3 (
        Device = hme0
        NetworkHosts = { "10.118.11.90" }
        )

    requires group cvm online local firm
    Db2_IP3 requires Db2_NIC3
    db2udb3 requires Db2_IP3

    // resource dependency tree
    //
    //          group db2mpp_grp3
    //          {
    //          Db2udb db2udb3
    //              {
    //              IP Db2_IP3
    //                  {
    //                  NIC Db2_NIC3
    //                  }
    //              }
    //          }
```

DB2 instance running in a Solaris zone

This sample DB2 instance running on Solaris gives the following configuration, which reflects a DB2 UDB instance running in a Solaris 10 zone environment.

```
include "types.cf"
include "/etc/VRTSagents/ha/conf/Db2udb/Db2udbTypes.cf"

cluster db2zone (
    UserNames = { "z_zoneres@vcs_lzs@sysA.engba.veritas.com" = Gn,
        "z_z1@vcs_lzs@sysA.engba.veritas.com" = aH }
    ClusterAddress = "10.178.6.32"
    SecureClus = 1
    CredRenewFrequency = 0
    CounterInterval = 5
    )

system sysA (
    )

system sysB (
    )

group Db2grp (
    SystemList = { sysA = 0, sysB = 1 }
    AutoStartList = { sysA, sysB }
    Administrators = { "z_zoneres@vcs_lzs@sysA.engba.veritas.com" }
    )

    DiskGroup z-dg (
        DiskGroup = db2dg1
        )

    IP ipres (
        Device = bge0
        Address = "10.178.6.28"
        ContainerName = "zone1"
        )

    Mount z-mnt (
        MountPoint = "/zones/db2data"
        BlockDevice = "/dev/vx/dsk/db2dg1/db2dg1data"
        FSType = vxfs
        FsckOpt = "-y"
        )

    NIC z-nic (
        Device = bge0
        NetworkType = ether
        NetworkHosts = { "10.178.2.4" }
        )
```

```
Volume z-vol (
    Volume = db2dg1data
    DiskGroup = db2dg1
    )

Zone zoneres (
    ZoneName = zone1
    )

Db2udb db2udb1 (
    ContainerName = "zone1"
    DB2InstOwner = "db2inst1"
    DB2InstHome = "/db2inst1"
    )

ipres requires z-nic
ipres requires zoneres
z-mnt requires z-vol
z-vol requires z-dg
zoneres requires z-nic
zoneres requires z-mnt
db2udb1 requires zoneres

// resource dependency tree
//
//          group Db2grp
//          {
//          IP ipres
//              {
//              NIC z-nic
//              }
//          Mount z-mnt
//              {
//              Volume z-vol
//                  {
//                  DiskGroup z-dg
//                  }
//              }
//          Zone zoneres
//              {
//              NIC z-nic
//              }
//          }
```

Index