High-Availability @ Novell
A Beigepaper

by Grettir Asmundarson (grettir@neticus.com)
Last Revised: February 9, 2000

This draft is in a state of roughness.
It is posted here for your inspection/comments, but please do not distribute it.

A Service Of Novell's Information Services & Technology Global Technical Architecture Group

About The "@ Novell" Series

Most documentation starts as hastily scrawled notes from sleep-deprived developers who weren't necessarily hired for their keen communication skills. Those notes are then fleshed out by recently graduated English majors who have spent their last four years immersed in works of fiction. The results are then passed on to the marketing department whose job it is to make sure that no word or phrase, even if it's true, will reflect unfavorably on the product ("I don't think that the word 'Basic' properly communicates the exciting nature of the product. Why don't we call it 'Visual Zesty!?!'"). It is then beset by lawyers who finish the job by making sure that they haven't explicitly promised that the product will actually do anything.

By the time the documentation gets into your hands, it has been so sanitized for your protection and generalized beyond recognition that you usually have to go out and buy a 3rd-party manual (that was, more likely than not, written by the same non-technical technical writer who wrote the original documentation) in a vain attempt to get an unbiased, unexpurgated, and/or unfiltered view of just how you're really supposed to use the stuff.

That's where the "@ Novell" series comes in. Rather than the vague, generalized, and wholly fictional examples found in most documentation, we're going to tell you exactly how we use our own products to run our own company. We are not, after all, a small, tidy computing environment suitable for documentation. We are a big, sprawling, untidy computing environment made up of over 500 production servers and 20,000 workstations in 130 locations throughout the world. In other words, we're probably an awful lot like you.

But please keep in mind that this document may be more than a little rough. It wasn't conceived by a committee, written by a committee, or approved by a committee, so it hasn't been edited, re-edited, tidied up, sanitized, and whitewashed. Don't think of this as an official whitepaper. It's more like a beigepaper.

The EUNUCH System

In the "ManageWise @ Novell" beigepaper, I outlined a fiendishly clever management system that I have devised called the End-User Notification & Ubiquitous Complaint Hotline (EUNUCH) System. It is a simple, effective, and inexpensive way to manage your network and services.

The End-User Notification & Ubiquitous Complaint Hotline System consists of two parts:

1. You sitting at your desk.

2. The end-user who calls you when something goes down.

No, really. Think about it. The advantages are obvious:

1. You are notified of any service outage in direct proportion to the importance of the service. If a service is very important, a large number of users will usually call and complain almost immediately. If the service isn't very important, you might get a call after a couple of hours…if at all. This allows you to instantly prioritize since it focuses your limited resources on service outages of the largest magnitude.

2. It is much less resource-intensive than active monitoring. There's no spending hours configuring SNMP traps, fine-tuning monitoring thresholds, and staring glassy-eyed at consoles. You can sit back, relax, and catch up on that stack of PC Weeks that has been collecting dust for the last three months, knowing that the EUNUCH System will kick in at the first sign of trouble.

It's brilliant. Now all I need is a couple of venture capitalists and a branding consultant.

Appalling…And Appallingly Widespread

Of course, such behavior is not only appalling…it is appallingly widespread. Some internal IS organizations, having what amounts to a monopoly on information services, start behaving like monopolists: overcharging, providing poor service, behaving like bullies, illegally tying their IS organization to the Accounting Department and saying that they can't be removed for fear of hurting consumers, etc.

But while you might be able to get away with such behavior with your internal customers (unless your company has an internal equivalent of the Justice Department), it simply doesn't work with external customers. Why? Because:

Customers don't call. Customers leave.

[Cut along dotted line and tape to computer monitor.]

And not only do customers leave, they usually don't come back. This tends to make CEOs very cross.

So, to avoid being outsourced faster than you can say "capital depreciation," you may need to reevaluate your current support policies and familiarize yourself with the Larry, Moe, & Curly of service outages: frequency, magnitude, and duration. Because when it comes to service outages, those three things are crucial.

Frequency How often does it happen? Rarely? Sometimes? Regularly? So often that vice presidents know you by name?

Magnitude To what extent are your customers affected by the service outage? Are they inconvenienced or incapacitated? What is the likelihood of your getting fired because of the outage?

Duration How long were your customers affected by the service outage? From the time the outage started to the time it was resolved, how many phone calls did you *not* pick up? 10? 100?

This beigepaper is not going to cover duration. That's something that should probably be covered in a separate "Disaster/Idiot Recovery @ Novell" beigepaper. But, in this beigepaper, we're going to talk about how we're working to increase the magnitude of our service outages while decreasing the frequency of them.

Increasing Magnitude

That's right, **we're actively working to *increase* the magnitude of our service outages**. That may sound ridiculous, but there is a method to the madness. (At least on paper…)

If we really wanted to *minimize* the magnitude of service outages, one of the best things we could do would be to employ the principle of dilution. If you only have one server and that one server goes down, that's pretty much a 100% service outage. But if you have ten servers and one of them goes down, that might only represent a 10% service outage. So, to fully leverage the principle of dilution, we should

probably maintain a 1:1 ratio of servers to users. That way, any single outage would only affect a single user.

But, instead, we are concentrating more and more people and services on fewer and fewer servers, which virtually guarantees that any single outage will have a much greater impact than ever before. So, why are we doing it?

One of the reasons that we are packing more people and services onto our servers is to make sure that we can actually do it. It would be presumptuous of us to make our customers test the capacity and scalability of our products.

But, more importantly (to the accountants, at least), it's cheaper. It costs much less for us to buy and maintain a single server with 500 users than it does to buy and maintain five servers with 100 users on each:

- It costs less for us to purchase the hardware initially.
- We get higher Transactions Per Minute (TPMs) and more disk space for every dollar spent.
- It takes up less space in our Data Center (which is very expensive real estate, indeed).
- It allows us to focus our limited support/maintenance/administration resources more effectively.
- It improves backup performance and reliability.
- It allows us to provide greater fault tolerance.
- It simplifies our disaster/idiot recovery capabilities.
- It thrills the Executive Briefing people to no end because it populates the Data Center with big boxes with lots of colorful flashing lights that look quite impressive to visiting executives on corporate tours who also tend to be fascinated by small, shiny objects.

This server consolidation will take place over the course of the next year or so and consists of the following:

- Investing in large, shared storage arrays.
- Investing in fewer, but more powerful, servers.
- Implementing large, SCSI tape libraries.
- Increasing the size of server clusters to take advantage of cost savings.
- Migrating tape libraries into the fibre-channel switch.

The large, shared storage arrays will allow us to save quite a bit of money up front. We can save almost 25% by purchasing ten servers that share a large storage array over ten standalone servers with equal disk space. We'll also get increased flexibility since we can dynamically allocate the shared storage to the servers where it's needed, rather than having unused disk space trapped in standalone servers.

So, since we're purposely doing nothing to minimize the potential magnitude[1] of any service outages, we'd better be doing something to decrease the frequency of those outages…and we are.

[1] This might sound scandalous but I'm not sure I *want* to minimize the magnitude of my service outages. My goal is to be indispensable. I want to be providing services that are of incalculable value to a large segment of the population at the lowest cost possible. If that is the case, then any service outage that I experience should have enormous magnitude, right? Give me an outage that brings the company to its knees and I'll glow with the satisfaction of a job well done. After all, if a service falls in the woods and nobody hears it, then why in the world are you providing that service in the first place?

Decreasing Frequency

When most people hear the phrase “high-availability,” they usually think of one thing: clustering. But while clustering certainly has a place in any high-availability solution, it’s just one part of the equation. It does little good to have a cluster running 24x7 if your customers can’t get to it because a backhoe just took out your Internet link or cheap hard drives in your expensive fibre-channel drive array just went south. You’ve got to take a holistic approach.

Hardware

Obviously, one of the most basic things you can do to decrease the frequency of service outages is to make the hardware on which the service runs as fault-tolerant as possible. Here @ Novell, we deploy the following on all of our production servers:

- ECC Memory
- Host Adapters With Battery-Backed Cache
- RAID 5 Drive Arrays
- Hot-Swappable Drives
- Predictive Failure Analysis

Servers hosting our more mission-critical services also get:

- Redundant Hot-Swappable Power Supplies
- Redundant Hot-Swappable Fans
- PCI Hot Plug

And we are starting to deploy:

- Redundant Host Adapters
- Redundant/Dual-Attached NICs

But we don’t bother with:

- Redundant Processor Power Modules (We’ve simply never had one fail…)

And this is just a partial list of what is available to you. As more and more high-availability features drift down from the mid-range/mainframe world, it’s getting easier and easier to run even your most valuable services on reliable, manageable, fault-tolerant, and relatively inexpensive Intel-based hardware.

Protected Memory

One of the real benefits of NetWare 5.x is its ability to run applications in protected mode. This means that if an application abends it doesn’t abend the server. It simply abends it’s own address space, leaving the OS and other applications intact.

We now have most of our large (>400 user) GroupWise Post Offices running in protected mode. As an example of how we’re doing this, we used to have a MAIL.NCF file on all of our Post Offices that looked like this:

MAIL.NCF

```
search add sys:\system\ngwnlm
load gwpoa /home-\\[server]\[volume]\[directory]
```

The transition to protected memory involved the rather tedious task of creating a second, two-line NCF file:

MAIL.NCF

```
protect gw.ncf
protection restart gw.ncf
```

GW.NCF

```
search add sys:\system\ngwnlm
load gwpoa /home-\\[server]\[volume]\[directory]
```

That's it. This loads GroupWise in protected mode and then tells the server to automatically restart GroupWise in a new address space if it abends.

As someone whose mailbox resides on the Post Office that is the first to run any alpha/beta code, I pledge allegiance to protected memory every day. It has done wonders to enhance my productivity[2].

Redundant Systems

Another way to decrease the frequency of service outages is to use a clustering solution like NetWare Cluster Services for NetWare 5. While this kind of failsafe solution may not be necessary in every situation, when you've got a truly mission-critical service that represents a single point of failure in a business process, it can be a Very Good Thing[3].

Let's say that you have two servers, one is running a Web Server and the other hosts a database that stores the information that is used by the Web Server. In this case, if one of the servers were to lose a power supply, the other server would be useless, too, and your whole Web site would be toast.

You could implement a solution like Novell StandbyServer, where you would have a duplicate, mirrored server for every server you wanted to protect. This is a great solution, but what if you can't afford to have two servers sitting around, twiddling their thumbs, waiting for the other servers to fail?

In the past I've always thought, "Why can't I just yank the hard drives out of the dead server, hot-plug them into my good server, mount the volumes on-the-fly, and restart the service?" Well, now you can, metaphorically speaking.

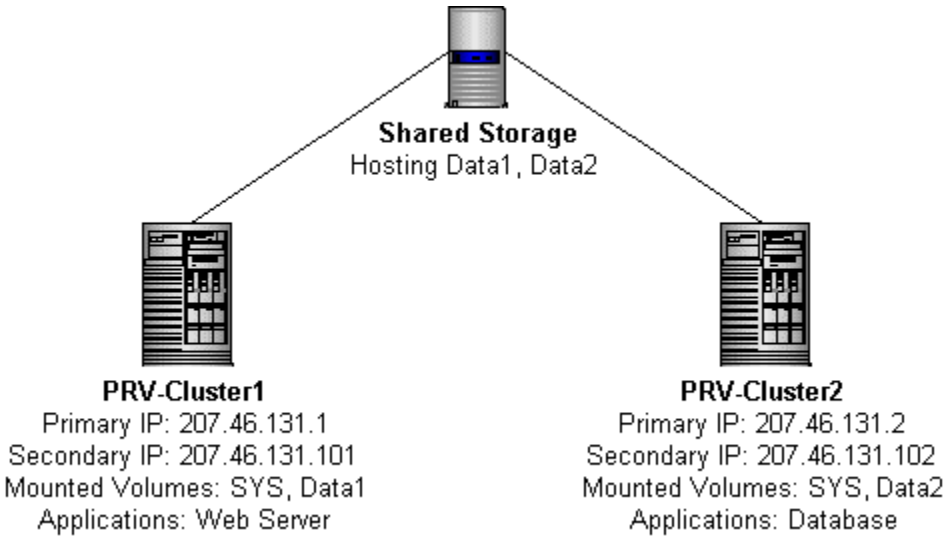
The concept is simple enough (though I'm sure the underlying code is not):

- Each server has its own unique IP address.
- Each service (Web server, database) has its own unique secondary IP address.
- Instead of hot-swapping hard drives you use a shared storage array to host the volumes that might need to be shared between the servers in the cluster.

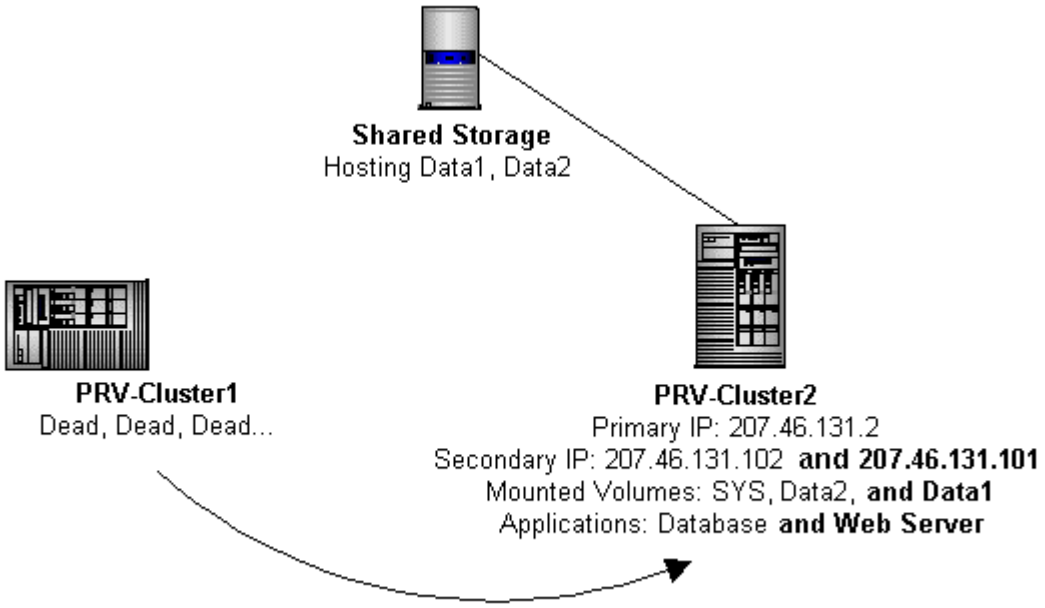
Here's an amateurish Visio diagram that illustrates this:

[2] In my case, "enhanced productivity" simply means that I can generate mediocre product at a much greater rate than ever before.

[3] And it does wonders for your uptime stats. If you need to upgrade the memory in all three servers in a cluster, you can simply migrate the applications on one server to another in the cluster, down the server, upgrade the memory, bring it back online, and migrate the applications back. In the past, such an upgrade would have caused three separate service outages, but now your uptime stats are unblemished.

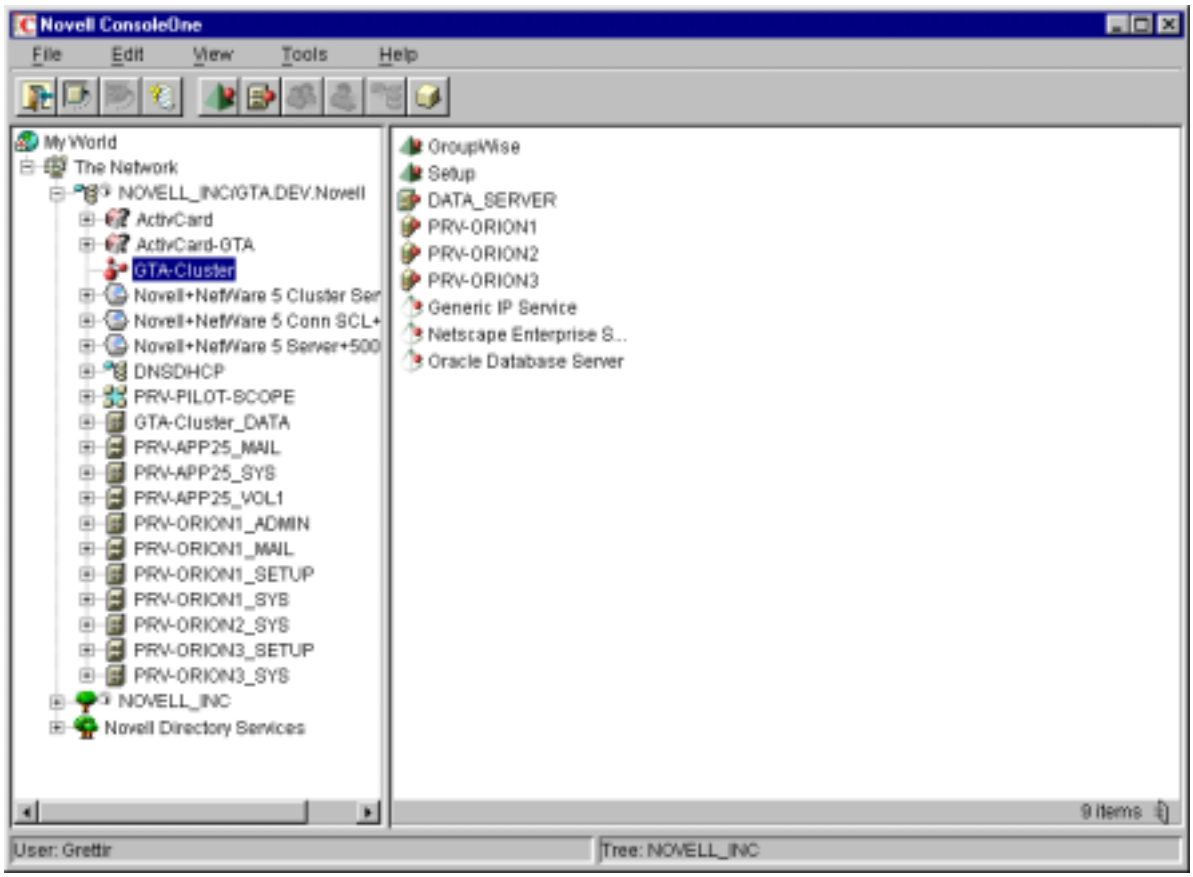


If PRV-Cluster1 failed, PRV-Cluster2 would mount the DATA1 volume, pick up the Web Server's old IP address, and restart the Web Server process:



…and it would happen so fast that your customers wouldn't even know anything had happened. You just need to be sure that, if all of the other servers in the cluster went down, the last remaining server would have the resources necessary to run all of the clustered applications. (At least long enough to get the other servers back on their feet.)

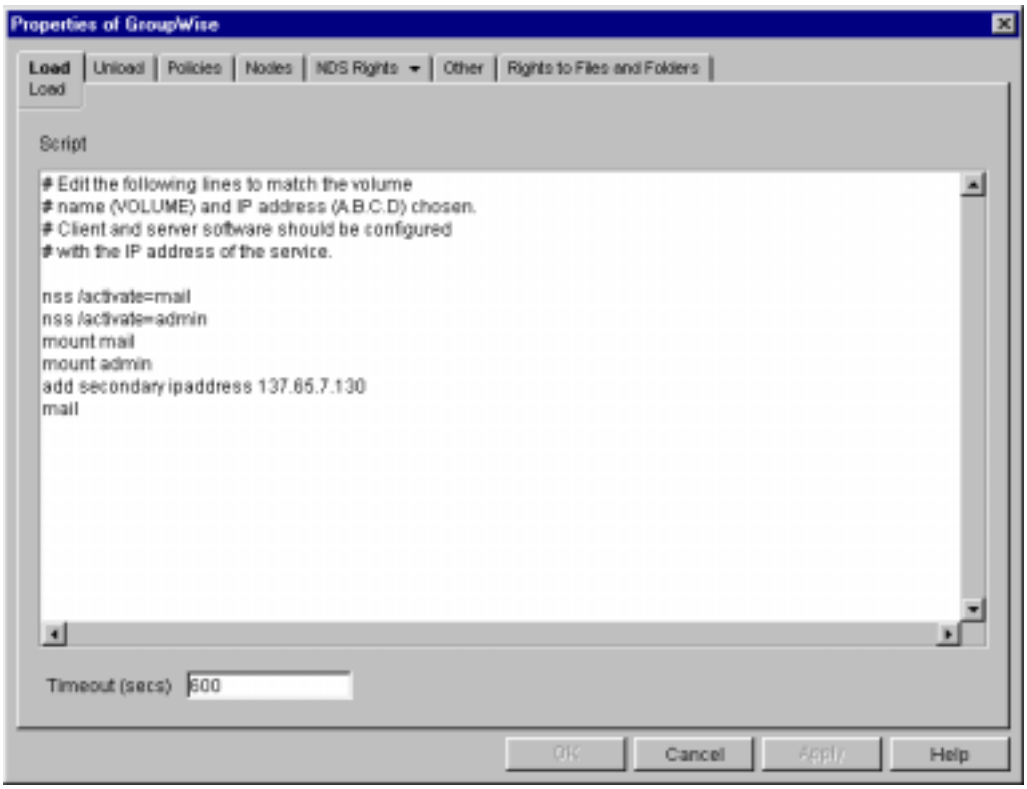
Configuring Cluster Server for NetWare 5 is also simple enough (though I'm sure the underlying code is not). Here's a rundown of the Cluster that we currently have running in our Lab:

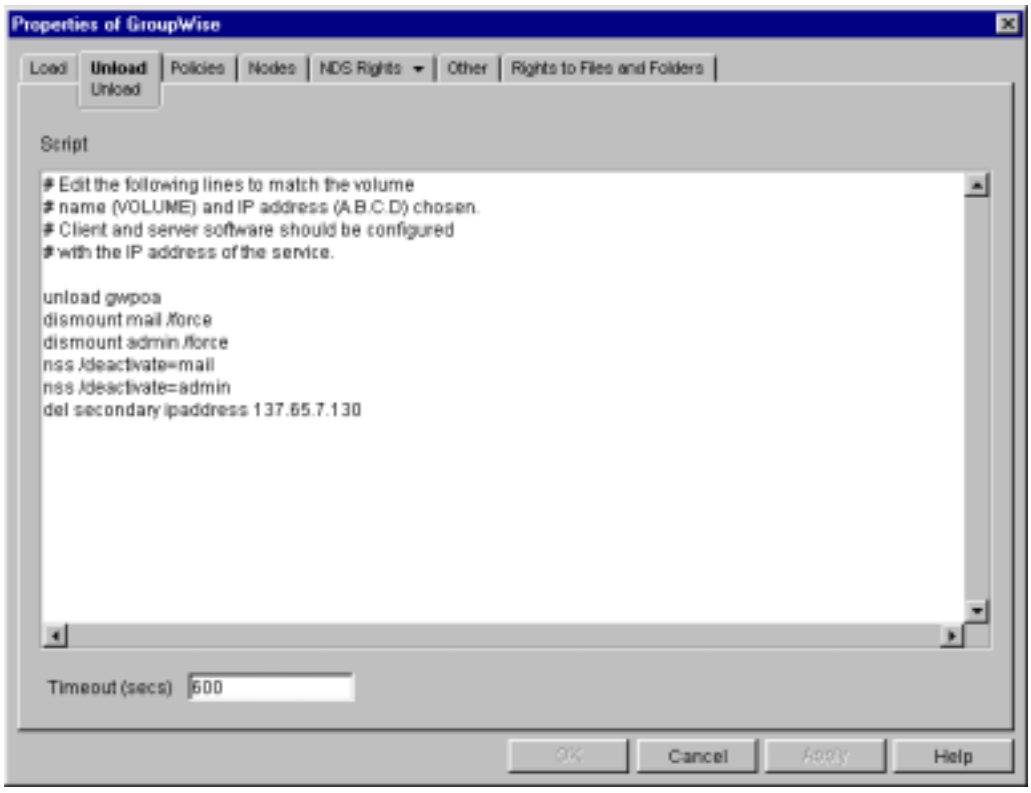


In this particular case, we have three servers in the cluster (PRV-ORION1, PRV-ORION2, and PRV-ORION3) and we are running two clustered applications: a GroupWise Post Office (GroupWise) and a Web Server hosting some of our Setup and configuration tools (Setup)[4].

Let's take a quick look at how we've set up the GroupWise "Cluster Resource" object. First, you specify the commands that should be run to load and unload the clustered application:

[4] This cluster is pretty sparsely populated since, now that NetWare Cluster Services has shipped, we've migrated most of the other services to our production environment. I just don't want you to presume from this particular example that you need to have one server in each cluster just twiddling its thumbs waiting for one of the other servers to go down. You can really have as many of the cluster servers cranking away on as many applications as you'd like.


Properties of GroupWise
Load
Unload
Policies
Nodes
NDS Rights
Other
Rights to Files and Folders
Load
Script
Edit the following lines to match the volume
name (VOLUME) and IP address (A.B.C.D) chosen.
Client and server software should be configured
with the IP address of the service.
nss /activate=mail
nss /activate=admin
mount mail
mount admin
add secondary ipaddress 137.65.7.130
mail
Timeout (secs)
600
OK
Cancel
Apply
Help


Properties of GroupWise
Load
Unload
Policies
Nodes
NDS Rights
Other
Rights to Files and Folders
Unload
Script
Edit the following lines to match the volume
name (VOLUME) and IP address (A.B.C.D) chosen.
Client and server software should be configured
with the IP address of the service.
unload gwpoa
dismount mail /force
dismount admin /force
nss /deactivate=mail
nss /deactivate=admin
del secondary ipaddress 137.65.7.130
Timeout (secs)
600
OK
Cancel
Apply
Help

If these commands look familiar, it's because these are essentially just NCF files. When we Load the service, we do the following:

```
nss /activate=mail
nss /activate=admin
mount mail
mount admin
add secondary ipaddress 137.65.7.130
mail
```

This:

1. Activates and mounts the partitions that hold the GroupWise code and data.
2. Binds the IP address for the GroupWise service to this server.
3. Runs the MAIL.NCF that loads the GroupWise Post Office in protected mode.

When we Unload the service, we basically do the same things in reverse:

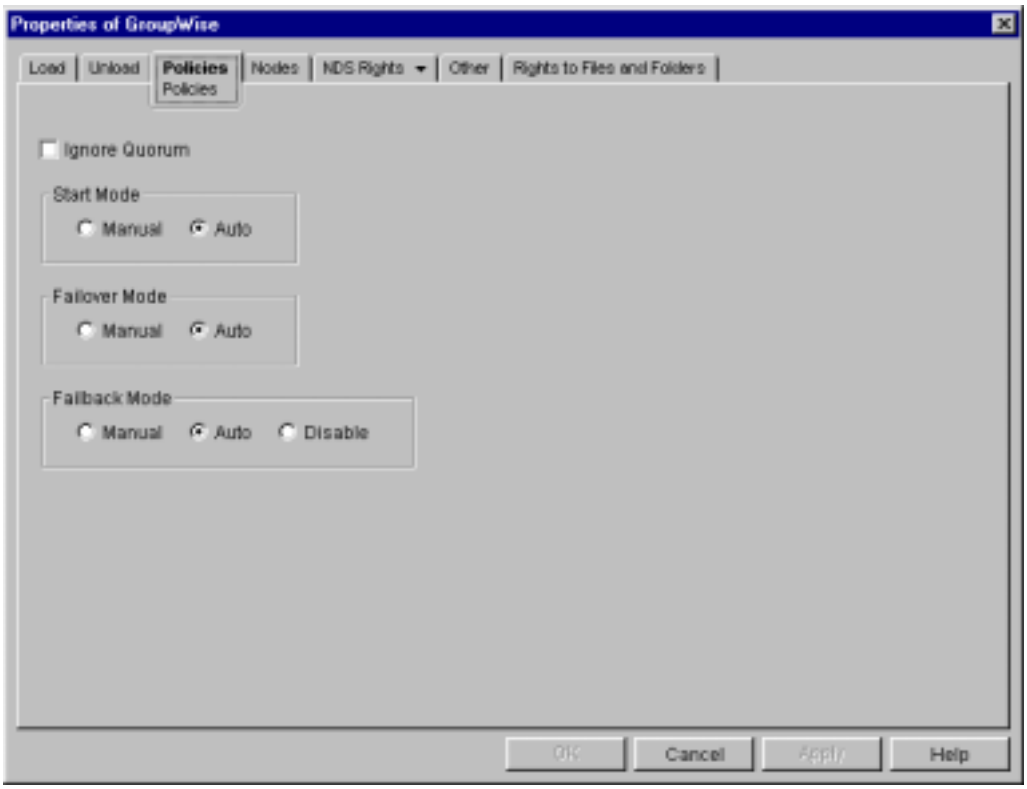
```
unload gwpoa
dismount mail /force
dismount admin /force
nss /deactivate=mail
nss /deactivate=admin
del secondary ipaddress 137.65.7.130
```

This:

1. Unloads the GroupWise Post Office.
2. Dismounts and deactivates the volumes that hold the GroupWise code and data.
3. Unbinds the IP address of the service.

Notice the use of the "**/force**" option on the **dismount** commands. If there are files open on a volume that you are trying to dismount, NetWare will inform you of this fact and then ask you whether you really want to dismount the volume. This is a great safeguard if you're manually downing the server, but the last thing you want is to have your non-stop application stopping and refusing to go on until someone types a "Y" at the console. The "**/force**" option avoids the prompt so that your non-stop application isn't stopped.

Next, you set up the policies for the process:

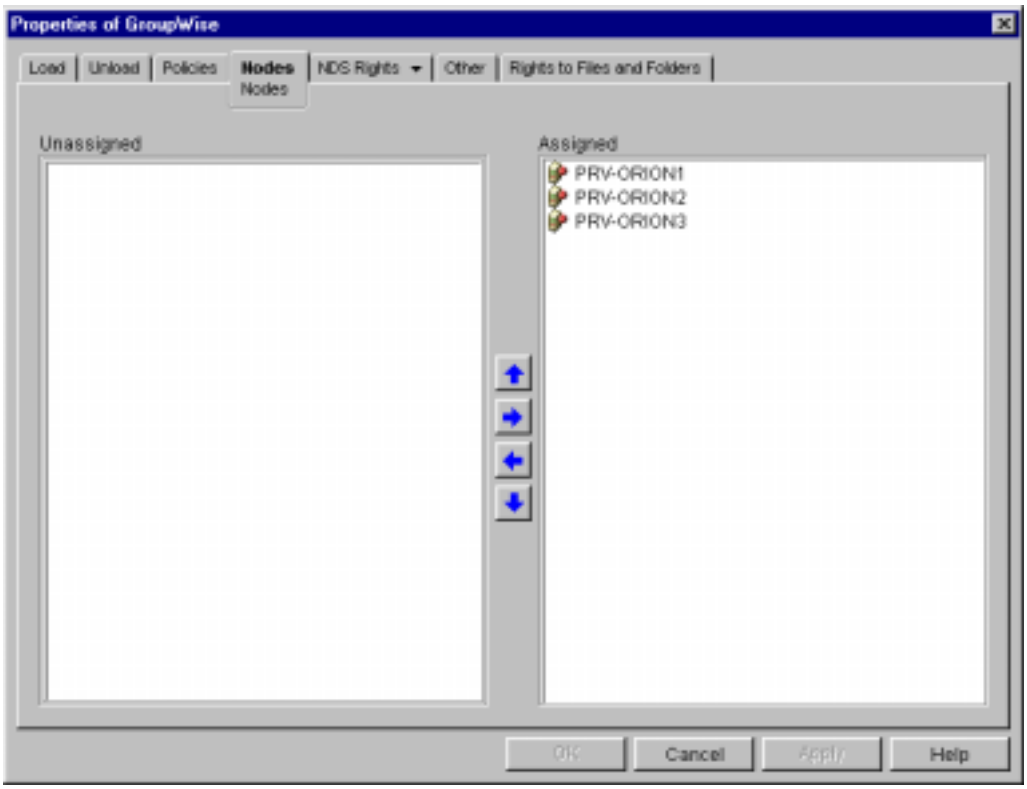


Do you want the application to start automatically when the cluster comes up, or do you want to start it manually each time?

Do you want the application to automatically fail over to another server in the cluster if something goes horribly wrong, or do you only want it to migrate when you tell it to?

And if the member of the cluster that died comes back to life, do you want the application to fail back automatically, manually, or not at all?

Then, you just tell the Cluster Resource which nodes it can run on:



In this case we want the GroupWise Post Office to be able to run on all three members of the cluster and we want PRV-ORION1 to be the first in line. In other words, when the cluster comes up, GroupWise will start on PRV-ORION1. If PRV-ORION1 is unavailable, the Post Office will fail over to PRV-ORION2 (or PRV-ORION3 if PRV_ORION2 is having problems, too). When PRV-ORION1 comes back up, the GroupWise Post Office would automatically fail back to it, since it is the first server in the list.

Here @ Novell, we're currently running a few GroupWise Post Offices, Oracle databases, and Netscape Web servers on some NetWare clusters scattered around the company, but this quarter we're making a big push to migrate some of our biggest mail and application servers into clusters.

Another example of a clustered solution is the "cache clustering" built in to our Novell Internet Caching System. We have a number of reverse proxies in our DMZ fronting www.novell.com. If one of them falls over another will simply pick up its IP address and service its requests along with its own. And if one of the Web servers hosting www.novell.com falls over, the reverse proxies are smart enough to recognize when they're not getting a response and will start filling from a different origin server.

Networking

And lastly, none of this does you any good if no one can get to your services in the first place, so a fault-tolerant networking infrastructure is just as important as the other pieces of the puzzle.

As far as our external connections are concerned, we have four DS3s coming in to our main site in Provo, UT, each capable of handling 45Mb/sec, and we use four different vendors for our four different DS3s. This complicates things to some degree, but we're less likely to suffer simultaneous catastrophic backhoe disasters on four DS3s from four different vendors at the same time.

We have two different routers handling the traffic on our DS3s. Two of our DS3s terminate at one router and two terminate at the other. If one router gives up the ghost, we're still functional.

We also try and maintain bandwidth equal to 2x our normal run rate. For instance, our current run rate is averaging 70Mb/sec. Therefore, we should have at least 140Mb/sec of capacity to handle unexpected surges in traffic and/or backhoe incidents.

Most of the connections to our outlying sites are at least 256k, either frame relay or dedicated lines. And at more and more sites we are implementing backup connections that are at least 50% of the standard connection. For instance, if a sales site has a 256k frame relay connection, we might install a 128k ISDN connection as a backup. These backup connections are tested at least once every quarter to ensure that they are still operational[5]. For added security, our critical sites have a permanent backup connection that is live at all times.

We've also built quite a bit of redundancy into our internal network infrastructure, as well, and we've started experimenting with Dual-Attached NICs in our servers. Dual-Attached NICs share a MAC address and IP address, but are attached to two different switches. If we loose one of the switches or NICs, we're still functional. And, as an added benefit, they can load balance between them.

Putting The Pieces Together

Now, after having said all of that, I will readily admit that we are not yet operating in the realm of some of the high-availability UNIX systems that are available. But I will also readily admit that I wouldn't be willing to invest the kind of money and resources necessarily to buy, maintain, and manage those behemoths.

Here @ Novell, by simply putting all of the appropriate pieces together, we're achieving over 99.8% uptime at a relatively low cost even though we're running some appallingly quirky alpha and beta code at times.

The fact is that we live in an imperfect world (which is painfully obvious to me each time I open my paycheck), but that doesn't stop our employers from expecting perfection, does it? This expectation of perfection isn't anything new, though:

> "American women expect to find in their husbands a perfection that
> English women only hope to find in their butlers."
>
> - W. Somerset Maugham (1874-1965)

Perhaps I should cluster myself. That way, any catastrophic failure on my part would transfer all of my responsibilities to someone else in the cluster.

Hmmm…

Acknowledgments

Nothing in this beigepaper represents original thought on my part. I couldn't have written a word without the generous help and input from everyone in Novell's IS&T Global Technical Architecture group. (I'd name them all individually but they'd probably get spammed.)

Send all comments, questions, corrections, and/or complaints to:

[5] We implemented the quarterly tests because of Pac Bell's rich history of stealing the IP addresses of our backup ISDN connections.

grettir@neticus.com

Tasty baked goods can be sent to:

Grettir Asmundarson
PRV-C-122
122 E. 1700 S.
Provo, UT 84606

And please note that Grettir Asmundarson is just a ridiculous pseudonym, so don't bother trying to call. You'll only confuse our receptionist.