Reviewing Transtec PROVIGO 410E iSCSI RAID

Last week I got myself a now toy at work: a Transtec PROVIGO 410E iSCSI RAID device.

The task was to review if it could serve its purpose as (expandable) external storage system for my company's (then still new-to-be-built, now in service) ftp mirror, a disk-based backup system and other possible areas of working.

This hardware was available for testing:

- 1 SuperMicro 6014P-T Server stocked with 2 Intel Xeon CPUs @ 3.0 GHz, 4 GB RAM, 2 x 160 GB SATA (internal RAID1), 2 Intel Pro/1000 adaptors (network connectivity) and 2 Intel Pro/1000MT adaptors for dedicated iSCSI storage network
- 1 Transtec PROVIGO 410E stocked with 1 Intel P4 CPU, 1 GB RAM, 15 x 400 GB SATA drives, single controller with 2 Broadcam Gigabit adaptors
- HP ProCurve J4903A Switch 2824 (for Gigabit Testing with Jumbo Frames)

This article will look into intial setup of the Provigo, especially step-by-step via serial console, which is not covered in the official manuals.
Furthermore connecting the device to a frontend server (single-host configuration via iSCSI, global file system not considered for now) running RedHat Enterprise Linux will be outlined.

#1 Getting Rid Of The Packaging

Initially, the device was ordered back in April 2007. Due to some stock shortage delivery was first scheduled for mid of May 2007 and was later re-scheduled for June.

When it arrived there came that incredibly huge box which had to be lifted around by two people because of its size and weight (~ 50 kg's).

Finally, after opening the box, I stood in front of the device. Along with it a pair of heavy rails, a serial cable and a documentation CD-ROM were included. Needless to say that a power cable with a german plug was shipped. Amazing that there are still some vendors who have not realized that we have a different power plug in Switzerland which is physically totally incompatible to the german ones.

Well, let's leave this aside for now and look into the configuration stuff.

#2 Preparing the Frontend Server

First things first, I prepared the frontend server with a clean minimum install of RedHat Enterprise Linux 5.0.
While the basic installation is clearly beyond the scope of this article, below are the additional steps I have gone through.

#3 Disable Xen Console

If you happen to have only one serial port and also a Xen-enabled kernel installed, the serial console must be released from Xen. This is done by adding **xencons=off** to the kernel appen line(s) in /etc/grub.conf like this:

```
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
```

```
title Red Hat Enterprise Linux Server (2.6.18-8.el5xen)
        root (hd0,0)
        kernel /boot/xen.gz-2.6.18-8.el5
        module /boot/vmlinuz-2.6.18-8.el5xen ro root=/dev/md0 xencons=off
        module /boot/initrd-2.6.18-8.el5xen.img
```

#3 Install minicom

For serial management a terminal emulator is required. I choose minicom which requires these rpm's to be installed (sample applies to x86_64).

```
rpm -i lockdev-1.0.1-10.x86_64.rpm  minicom-2.1-3.x86_64.rpm
```

If your RHEL is configured to use a local repository via yum, you may run this instead:

```
yum install minicom-2.1-3.x86_64
```

To configure minicom run:

```
minicom -s
```

This will enter the setup dialog where I choose **serial port setup** first.

The Provigo has a 115200 8N1 factory default for the serial console, so I changed my settings as shown below.
I assumed /dev/ttyS0 (COM1) from my current hardware configuration.

```
â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â?•
   â?? A -    Serial Device      : /dev/ttyS0                    â??
   â?? B - Lockfile Location     : /var/lock                     â??
   â?? C -   Callin Program      :                               â??
   â?? D -  Callout Program      :                               â??
   â?? E -    Bps/Par/Bits       : 115200 8N1                    â??
   â?? F - Hardware Flow Control : No                            â??
   â?? G - Software Flow Control : No                            â??
   â??                                                           â??
   â??    Change which setting?                                  â??

â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??â??
```

Back to the main menu I entered **modem and dialing** where I removed the init strings so no AT init commands are sent to the host accidentally.

Afterwards I choose **save setup as dfl** and **exit from minicom**.

#4 Configure the Provigo

To configure the Provigo attach the serial cable shipped with the device to the RJ11 jack on the Provigo and the RS232 connector on the frontend host.
No start minicom and press enter. If everything works out you should receive a login prompt.

```
Welcome to minicom 2.1
```

OPTIONS: History Buffer, F-key Macros, Search History Buffer, I18n
Compiled on Jul 26 2006, 06:38:09.

Press CTRL-A Z for help on special keys

login:
The login name is **root** with a default password of **root**.

You may do the initial setup by running the 'dasetup' command for Q&A based setup.
This will however only cover some basic settings like IP address and hostname. Advanced configuration is still to be done manually.
For this reason I'll apply all configuration manually.

First of all, set some arbitary hostname, a name server and - if required - a default route.

dahostname SAN001
dans -a 192.0.2.2
daroute add default gw 192.0.2.1
Now the network interface must be configured for either default a mtu of 1500...

daifconfig local da0 192.0.2.11 netmask 255.255.255.0 mtu 1500
********** local diskarray controller ID 0 **********
Set local host interface da0 192.0.2.11 netmask 255.255.255.0 mtu 1500
****
or a mtu of 9000 aka jumbo frames.

daifconfig local da0 192.0.2.11 netmask 255.255.255.0 mtu 9000
********** local diskarray controller ID 0 **********
Set local host interface da0 192.0.2.11 netmask 255.255.255.0 mtu 9000
****
Even if source addresses can be spoofed eventually, adding an IP access control list is never wrong.
So this will first remove the default acl and add the test network and loopback range to the access list:

daacl -d 0.0.0.0/0
daacl -a 192.0.2.0/24
daacl -a 127.0.0.0/8
And don't forget to set new passwords for **guest** (unprivileged login) and **root** (admin login).

dapasswd -p newpasswd guest
dapasswd -p newpasswd root
Mind that the **dapasswd** command does not support single and double quotes.
If you have a password which includes special characters, it must be set like this:

dapasswd -p my!special:32 root
You should also set the time zone and time/date of the device.
Refer to **dadate -h** for the input format.

dadate -z UTC +2
dadate -s 0702152607.00
Optionally email notification can be enabled as seen in **daaddrbk -h** command.

#5a Create Custom iSCSI Targets

Now that the basic configuration is complete it's about time to define our custom iSCSI targets.
In client-server terminology, an iSCSI target refers to the server-side, which exports a block device to a client.

Some basic rules for iSCSI target names, which are also known as IQNs (iSCSI qualified names), define that an IQN consists of multiple tokens separated by dots and colons.

1st part is always the keyword 'iqn'
2nd part equals to year and month when the domain name (see also 3rd part) was acquired, eg. 2004-07
3rd part is the reversed domain name, e.g. phunsites.net becomes net.phunsites
4th part is a colon as delimiter
5th part is a string or serial number which should refer to the storage system, eg. mac address, hostname, etc

So when getting this altogether a valid IQN could read like this:

iqn.2004-07.net.phunsites:san001
If you do now own a domain name or want to be strictly internal something like this could also be used:

iqn.2007-07.localnet.localhost:storage1
After having defined the proper IQN string it can be set and verified as follows:

datarget -set iqn.2004-07.net.phunsites:san001
Target names are set for both controllers.

datarget -show
Target name (local): iqn.2004-07.net.phunsites:san001.0
Target name (remote): iqn.2004-07.net.phunsites:san001.1

#5b Configure Disk Groups

Let's have a look into disk groups, which are actually the RAID sets.
Disk groups may be created from all empty/unassigned disks. List them by means of the **dadg** command.

dadg -Ldisk
Drive View from controller 0 View from controller 1
Slot 0 (xda) .....SPARE
Slot 1 (xdb) .....SPARE
Slot 2 (xdc) .....SPARE
Slot 3 (xdd) .....SPARE
Slot 4 (xde) .....SPARE
Slot 5 (xdf) .....SPARE
Slot 6 (xdg) .....SPARE
Slot 7 (xdh) .....SPARE
Slot 8 (xdi) .....SPARE
Slot 9 (xdj) .....SPARE
Slot 10 (xdk) .....SPARE
Slot 11 (xdl) .....SPARE
Slot 12 (xdm) .....SPARE
Slot 13 (xdn) .....SPARE
Slot 14 (xdo) .....SPARE

To create three disk groups with five disks per group forming a RAID5 use these commands:

#dadg -a DG1 -o 0 -l 5 -v 95 -m "Disk Group 1" /dev/xd[a-e]
#dadg -a DG2 -o 0 -l 5 -v 95 -m "Disk Group 2" /dev/xd[f-j]
#dadg -a DG3 -o 0 -l 5 -v 95 -m "Disk Group 3" /dev/xd[k-o]
Then verify how it looks like:

dadg -Lall

Information for all the disk groups in controller 0:
Disk group "DG1": active (vg_510842bddb23d1ed44cab1b242cdea52)
Disk group "DG2": active (vg_e944dd0ebcb7427aeea37cefdfc9c895)
Disk group "DG3": active (vg_a0e7be409536a4dd195cbead5986416b)

Information for all the disk groups in controller 1:

dadg -Ldisk
Drive View from controller 0 View from controller 1
Slot 0 (xda) .....IN DG ("DG1")
Slot 1 (xdb) .....IN DG ("DG1")
Slot 2 (xdc) .....IN DG ("DG1")
Slot 3 (xdd) .....IN DG ("DG1")
Slot 4 (xde) .....IN DG ("DG1")
Slot 5 (xdf) .....IN DG ("DG2")
Slot 6 (xdg) .....IN DG ("DG2")
Slot 7 (xdh) .....IN DG ("DG2")
Slot 8 (xdi) .....IN DG ("DG2")
Slot 9 (xdj) .....IN DG ("DG2")
Slot 10 (xdk) .....IN DG ("DG3")
Slot 11 (xdl) .....IN DG ("DG3")
Slot 12 (xdm) .....IN DG ("DG3")
Slot 13 (xdn) .....IN DG ("DG3")
Slot 14 (xdo) .....IN DG ("DG3")
You may also get specific information about any existing disk group.

dadg -Ldg DG1
Disk-Group Name : DG1
Status : active
Raid status : healthy
Owner controller : 0
Alarm level : 95
Raid Level : 5
Desired number of disks : 5
Actual number of disks : 5
Member disks : xda(0), xdb(1), xdc(2), xdd(3), xde(4)
DG size : 1525248 MB
Allocated size : 0 MB
Free size : 1525248 MB
Comment : Disk Group 1
#5c Create Volumes

Volumes are created from disk groups. Each disk group can host multiple volumes which will be exported to either one single host at a time or - by using an abstraction layer like global filesystem - to multiple hosts at once.

Volumes are maintained by the the **davd** command. Refer to **davd -h** for details on the arguments.

davd -a DG1VOL1 -g DG1 -m 2 -r 0 -s 1525248
DG1VOL1 successfully created and available.

davd -a DG2VOL1 -g DG2 -m 2 -r 0 -s 1525248
DG2VOL1 successfully created and available.

davd -a DG3VOL1 -g DG3 -m 2 -r 0 -s 1525248
DG3VOL1 successfully created and available.

davd -L DG1VOL1 -g DG1
Virtual-Disk Name : DG1VOL1
Type : Data Virtual Disk
Date/Time Created : Mon Jul 2 15:45:43 2007
Operational State : Available
Disk Group : DG1
Cache Policy : Write Back
Read Ahead Policy : Enabled
Capacity : 1525248 MB

#5d Add Initators

To allow any arbitary host to connect it must be added to the Provigo initiator list.

Again in client-server terms, the initator refers to the client connecting the the target (server).

The **dahost** command servers that purpose and will also allow to use initiator secrets for additional security.
I omitted the latter one for the sake of simplicity.

dahost -a -w iqn.2004-07.net.phunsites:testhost -n testhost.phunsites.net -t "Test Host for iSCSI" -b 0 -o 1

#5e Export LUNs to Initators

Now that our initiators are known to the system, the Volumes must be exported to the iniators.
For that purpose the **dalun** is used, which will assign a LUN (logical unit numbers) for each volume.

dalun -a -n testhost.phunsites.net -g DG1 -k DG1VOL1
dalun -a -n testhost.phunsites.net -g DG2 -k DG1VOL1
dalun -a -n testhost.phunsites.net -g DG3 -k DG1VOL1

#6 Configure Networking

The Provigo manual points out multiple possibilities on network connections which include variants of channel bonding and different MTU sizes.

These may or may affect the effective transfer rate for all frames, however this depends heavily on the usage of the iSCSI resource and how the data looks like (e.g. many small file transfers or less big file transfers).

If you want to use jumbo frames (MTU 9000) you will need to alter your **/etc/sysconfig/network-scripts/ifcfg-ethX** file. Add an MTU statement to set the value for the new MTU size.

```
# Intel Corporation 82546GB Gigabit Ethernet Controller
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
IPADDR=192.0.2.21
NETMASK=255.255.255.0
MTU=9000
```

If you want to try your luck with channel bonding, add this to **/etc/modprobe.conf**:

```
alias bond0 bonding
options bond0 mode=1 miimon=100
```

This will enable active-backup mode, which usually works best.
Refer also to **README.bonding**, which should be somewehre on your systems, for other bonding modes.

If your bonding devices includes eth2, your network configuration should read like this for **/etc/sysconfig/network-scripts/ifcfg-eth2**:

```
DEVICE=eth2
ONBOOT=yes
HWADDR=00:07:e9:1f:bc:08
USERCTL=no
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

And for **/etc/sysconfig/network-scripts/ifcfg-eth3** eventually:

```
DEVICE=eth3
ONBOOT=yes
HWADDR=00:07:e9:1f:bc:09
USERCTL=no
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

The network configuration goes to **/etc/sysconfig/network-scripts/ifcfg-bond0**:

```
DEVICE=bond0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
IPADDR=192.0.2.21
NETMASK=255.255.255.0
```

Again you may include the **MTU=9000** option to enable jumbo frames.

To bring up the bonding interface manually use this for example:

```
# modprobe bonding mode=1
```

ifconfig bond0 192.0.2.21 netmask 255.255.255.0 mtu 9000
ifenslave bond0 eth2
ifenslave bond0 eth3
#7 Configure iSCSI Initiator

Now the frontend host needs to be configured.
First of all installation of an iSCSI initiator software is required.

On RedHat Enterprise Linux 5.0 this can be installed as easy as:

rpm -i iscsi-initiator-utils-6.2.0.742-0.5.el5.x86_64.rpm
or

yum install iscsi-initiator-utils.x86_6
Then make sure the services are enabled properly, but not yet started.

chkconfig iscsid on
chkconfig iscsi on
chkconfig --list iscsid
chkconfig --list iscsi
service iscsid stop
service iscsi stop
Add the initiator name to **/etc/iscsi/initatorname.iscsi**. This should correspond to the initator name used earlier with the **dahost** command and also comply to existing host name entries in the dns zone if they also exist.

InitiatorName=iqn.2004-07.net.phunsites:testhost
Then the **iscsid** daemon may be started at first.

service iscsid start
Turning off network shutdown. Starting iSCSI daemon: [OK]
Run a discovery against the iSCSI target's IP address or hostname.
This should reveal the iSCSI target and bind a persistent connection to it.

iscsiadm -m discovery -t sendtargets -p 192.0.2.11
192.0.2.11:3260,1234 iqn.2004-07.net.phunsites:san001.0

iscsiadm -m discovery
192.0.2.11:3260 via sendtargets

iscsiadm -m node
192.0.2.11:3260,1234 iqn.2004-07.net.phunsites:san001.0
Then the **iscsi** service may be started:

service iscsi start
Turning off network shutdown. Starting iSCSI daemon: [OK]
Setting up iSCSI targets: Login session [192.0.2.11:3260 iqn.2004-07.net.phunsites:san001.0]
If everything went well something like this should arise in the system logs:

dmesg
scsi1 : iSCSI Initiator over TCP/IP

Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00
Type: Direct-Access ANSI SCSI revision: 04
SCSI device sdb: 3123707904 512-byte hdwr sectors (1599338 MB)
sdb: Write Protect is off
sdb: Mode Sense: 17 00 00 00
SCSI device sdb: drive cache: write back
SCSI device sdb: 3123707904 512-byte hdwr sectors (1599338 MB)
sdb: Write Protect is off
sdb: Mode Sense: 17 00 00 00
SCSI device sdb: drive cache: write back
sdb: unknown partition table
sd 1:0:0:0: Attached scsi disk sdb
sd 1:0:0:0: Attached scsi generic sg3 type 0
Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00
Type: Direct-Access ANSI SCSI revision: 04
SCSI device sdc: 3123707904 512-byte hdwr sectors (1599338 MB)
sdc: Write Protect is off
sdc: Mode Sense: 17 00 00 00
SCSI device sdc: drive cache: write back
SCSI device sdc: 3123707904 512-byte hdwr sectors (1599338 MB)
sdc: Write Protect is off
sdc: Mode Sense: 17 00 00 00
SCSI device sdc: drive cache: write back
sdc: unknown partition table
sd 1:0:0:1: Attached scsi disk sdc
sd 1:0:0:1: Attached scsi generic sg4 type 0
Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00
Type: Direct-Access ANSI SCSI revision: 04
SCSI device sdd: 3123707904 512-byte hdwr sectors (1599338 MB)
sdd: Write Protect is off
sdd: Mode Sense: 17 00 00 00
SCSI device sdd: drive cache: write back
SCSI device sdd: 3123707904 512-byte hdwr sectors (1599338 MB)
sdd: Write Protect is off
sdd: Mode Sense: 17 00 00 00
SCSI device sdd: drive cache: write back
sdd: unknown partition table
sd 1:0:0:2: Attached scsi disk sdd
sd 1:0:0:2: Attached scsi generic sg5 type 0
The device should then also be visible in **/proc/scsi/scsi**:

cat /proc/scsi/scsi
Host: scsi1 Channel: 00 Id: 00 Lun: 00
Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00
Type: Direct-Access ANSI SCSI revision: 04
Host: scsi1 Channel: 00 Id: 00 Lun: 01
Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00
Type: Direct-Access ANSI SCSI revision: 04
Host: scsi1 Channel: 00 Id: 00 Lun: 02
Vendor: Transtec Model: PROVIGO1100-SAN0 Rev: 1.00

Type: Direct-Access ANSI SCSI revision: 04

#8 Working With Disks

Now you can work with the disks as if they were locally installed to the system.
They are initialized like any other disk device would be, too:

```
# fdisk /dev/sdb
# fdisk /dev/sdc
# fdisk /dev/sdd

# mkfs -t ext3 /dev/sdb1
# mkfs -t ext3 /dev/sdc1
# mkfs -t ext3 /dev/sdd1
```

If you want the iSCSI block devices to be mounted via fstab on startup, the entry should include the **_netdev** keyword.
This will ensure that the network is available before the device is to be mounted.

```
/dev/sdc1 /mnt/sdc1 ext3 _netdev 0 0
```

#9 Conclusions

While I'm still hacking up some good benchmark scripts, I can tell for sure that the system performs very well and definitily fits my purpose.
As this is the first iSCSI based device I've ever got, implementation and usage is very simple and works reliable.

I'm in doubt however wether an iSCSI device is good enough when it comes to high-load scenarios where every bit of I/O counts.
This is definitely one thing I want to test out in-depth during the next weeks.

Despite still in evaluation, this review has already left its markings on the wall.
There were some odd's and evens which I came along and as such are to be mentioned here.

First, and already mentioned earlier, is the fact why I didn't get a power cable with a swiss power plug.
It _may_ actually sound pedantic, but when ordering anything from a vendor in Switzerland I do expect to receive to correct equipment.

My second thoughts care about the documentation.
I must admit that the guys at Transtec did a very good job in writing the handbook. It is very detailed and covers almost everything one needs to know.
It lacks however a step-by-step description how setup is done via the serial console.
While all commands are properly documented, one get's no idea whatsoever in which order they must be run.
It's a matter of puzzling it together by logical evaluation and a bit of trial and error.
One might argue that the docs cover step-by-step configuration for the web management interface. Acknowledged. This exists and is indeed very detailed and to the point.
But still, I know a lot of poeple who don't trust in web interfaces (me included). Providing a little check list would be sufficient enough but still of great help to get things done faster.

The third thing noted is the inconsequent usage of command line arguments within the CLI.
This can be seen for example when it comes to reviewing settings.

```
dadg -Lall (List)
dahost -l (list)
dalun -p (print)
```

daroute (nothing required at all to 'list')
As the **da*** commands are clearly part of the Provigo firmware, I can't follow the reason why the all have different syntax.
It would be more straight-forward if they'd follow a common guideline and define the same arguments for particular functions to be identical accross all utilities.

To make things even worse in the current implementation, there exist also variations which require the admin to _know_ in advance what he is querying for. An example of this:

davd -L DG1VOL1 -g DG1
davd is unable to show all defined volumes at once, there's simply no option for this. So I _must_ know the exact names of my volumes. But where do I get them from if I happen to forget the volume names one day?

Also the error reporting could be better or clearer.
What would this error message mean:

dalun -a -n hostXYZ -g DG1 -k DG1VOL1
Host not found
Host not found. Errr... First thing coming into my mind: host not found. Where? In DNS. In /etc/hosts? Where!?
Making the error message say something like 'Please add Host "HostXYZ" with "dahost" command first' would make things much more obvious.

Fourth thing to note is protocol support in Provigo.
It supports beneath http protocol also the serial console for management and ... telnet.
Huh!? telnet? Telnet to be used for management purpose looks like an ancient dinosaur to me!? Guys, we have 2007!
Even if an iSCSI-based storage network is supposed to be separated from public networks and therefore be closed down, using telnet for remote management is neither state-of-the-art nor secure.
I see no reason why Provigo could not support SSH. It has plenty of RAM, a decent CPU. Also the base OS image, which is build on some Linux, is around ~52 MiB with ~48 MiB free space, so room enough to include the SSH daemon. Why don't you do so?

A fifth thing sawn was a bug in **dastat** command:

Name: SAN001-0
Controller Slot#: 0
Remote Controller: Not Installed
Operational State: Active/-
Total Capacity: 4468.50 GB
Time: Mon Jul 2 16:45:33 UTC+2 2007
Kernel Version (local): K:2.7.155-20061114
CPLD Board Midplane Rev (local): C:7 B:A04 M:2
BIOS Version (local): RN2_0110 05/23/06
/usr/sbin/dastat: [: !=: unary operator expected
Fibre Channel WWN:

Drive View from controller 0 View from controller 1
Slot 0 (xda) .....IN DG ("DG1")
Slot 1 (xdb) .....IN DG ("DG1")
Slot 2 (xdc) .....IN DG ("DG1")
Slot 3 (xdd) .....IN DG ("DG1")
Slot 4 (xde) .....IN DG ("DG1")
Slot 5 (xdf) .....IN DG ("DG2")

Slot 6 (xdg) .....IN DG ("DG2")
Slot 7 (xdh) .....IN DG ("DG2")
Slot 8 (xdi) .....IN DG ("DG2")
Slot 9 (xdj) .....IN DG ("DG2")
Slot 10 (xdk) .....IN DG ("DG3")
Slot 11 (xdl) .....IN DG ("DG3")
Slot 12 (xdm) .....IN DG ("DG3")
Slot 13 (xdn) .....IN DG ("DG3")
Slot 14 (xdo) .....IN DG ("DG3")

No System Health Alert
Messages like **[: !=: unary operator expected** look definitely not good, especially when it's simply a matter of proper shell scripting syntax.
When looking at **/usr/bin/dastat** there is this line:

if [$multiple_target != "yes"]
Changing it as follows removes the error message:

if ["$multiple_target" != "yes"]
Despite these things, which could be regarded as minor glitches, I must admit that the Provigo 410E does an excellent job for a decent price.
Given it's modular design it's scalable and can fit multiple scenarios from single iSCSI environments up to mixed iSCSI and FC environments.

As mentioned before I'm doing some benchmarking currently, which I hope to reveal some interesting information in terms of throughput and I/O performance.