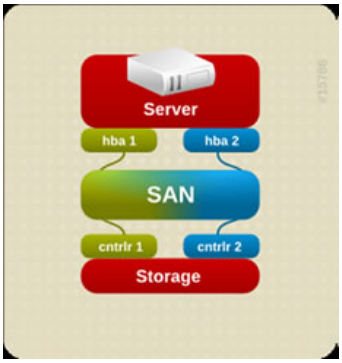
Red Hat Multipathing

This is a short document describing the Device Mapper Multipath (DM-Multipath) which is a feature of Red Hat Linux REL 5. DM-Multipath allows you to configure multiple I/O paths from your server to your storage device (SAN, etc). These are physical paths that include HBA, cables and switches. Multipathing aggregates the I/O paths, creating a new device that consists of the aggregated paths.



DM-Multipathing can provide the following

Redundancy	DM-Multipathing can provide failover capabilities, this can be configured in two ways • active/passive - only half of the paths are used • active/active - all the paths are used and used in a round-robin fashion.
Improved performance	DM-Multipathing can detect loading on the I/O paths and dynamically re-balance the load.

DM-Multipathing by default has support for most of the common SAN arrays, all devices supported are stored in a file *multipath.conf.defaults*. You can add support into the multipath configuration file */etc/multipath.conf* but more on this alter.

The following are the components that make up the multipathing

Kernel components	
dm-multipath kernel module	reroutes I/O and supports failover for paths and path groups
multipathd daemon	Monitors paths and takes action when necessary (failed paths, restored paths)
Multipath Configuration Files	
multipath.conf	The main configuration file, this is located in /etc.
multipath.conf.defaults	Lists support storage arrays, if your array is not listed it still may be possible to configure it in the multipath.conf file . The location of the file is /usr/share/device-mapper-multipath-???? depending on the version you have.
bindings	This file is located in /var/lib/multipath and can be used to define user-friendly names and make devices consistent across nodes.
Multipath commands	
multipath	List and configures multipath devices
kpartx	Creates device mapper devices for partitions on a device

Each multipath device has a World Wide Identifier (WWID), which is guaranteed to be unique and unchanging (I will discuss DR later which means that the WWID will change). By default the name of multipath device is set to its WWID but there is a option called **user_friendly_names** which sets the alias to a node-unique name of the form of **mpath*n***.

For example a server has two additional SAN LUN's connected to the server the outcome may be as follows

device name	WWID	dm-multipath names
/dev/sdb	20017380006c00009	/dev/mapper/mpath1 /dev/mpath/mpath1 /dev/dm-1
/dev/sdc		
/dev/sdd	20017380006c0000a	/dev/mapper/mpath2 /dev/mpath/mpath2 /dev/dm-2
/dev/sde		

/dev/sdb and /dev/sdc are in fact the same disk (same WWID), thus multipath will create one unique name mpath1, if either of the paths fail the server will be unaffected.

As you can see multipath creates three different ways to access the device

/dev/mapper/mpath*n*	These are create early in the boot sequence, thus these are ideal for logical volumes, boot devices
/dev/mpath*n*	are provided as a convenience so that all multipathed devices can be seen in one directory. These devices are created by the **udev** device manager and may not be available during startup.
/dev/dm-*n*	these are for external use only, however when I installed a fresh Red Hat these were used?

You also have a fourth option and that is to set an alias in the **multipath.conf** file. A multipath device has a number of attributes which can be changed in the **multipath.conf** file, we will see some examples of this later.

DM-Multipath setup

To setup DM-Multipathing the procedure is as follows

1. Install the **device-mapper-multipath** rpm (if not already installed)
2. Edit the **/etc/multipath.conf** configuration file
   - comment out the default blacklist or create you own exclude blacklist
   - change any of the default (if required)
   - save the configuration file
3. Start the multipath daemons
4. Create the multipath device with the **multipath** command

A basic **multipath.conf** file is below,

- the default section as mentioned above configures the multipath to use friendly names, there are a number of other options that can be used.
- the blacklist section excludes specific disks from being multipathed, notice the exclusion of all wwid disks
- the blacklist exceptions section includes the devices with a specific wwid to be included
- the multipath's section creates aliases that match a specific disk to a alias using the wwid

multipath.conf (basic)

```
defaults {
   user_friendly_names yes
   path_group_policy failover
}

blacklist {
   devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
   devnode "^(hd|xvd|vd)[a-z]*"
   wwid "*"
}

# Make sure our multipath devices are enabled.

blacklist_exceptions {
   wwid "20017380006c00034"
   wwid "20017380006c00035"
   wwid "20017380006c00036"
   wwid "20017380006c00037"
}

multipaths {
  multipath {
      wwid "20017380006c00034"
      alias mpath0
  }
  multipath {
      wwid "20017380006c00035"
      alias mpath1
  }
  multipath {
      wwid "20017380006c00036"
      alias mpath2
  }
  multipath {
      wwid "20017380006c00037"
      alias mpath3
  }
}
```

Once you have configured your multipath.conf, then perform the below to start multipathd

1. modprobe dm-multipath
2. service multipathd start
3. multipath -d (this will perform a dry to make sure you are happy with everything, fix anything that appears as a problem)
4. multipath -v2 (commits the configuration)
5. multipath -ll
6. chkconfig multipathd on (make devices are configured after a reboot)

Hopefully you should now a list similar to below, each device is active and ready.

multipath -ll (simple)

```
## This example shows my connections to a IBM XIV SAN, yours may look different

server1> multipath -ll |grep mpath
mpath2 (20017380006c00036) dm-7 IBM,2810XIV
mpath1 (20017380006c00035) dm-6 IBM,2810XIV
mpath0 (20017380006c00034) dm-5 IBM,2810XIV
mpath3 (20017380006c00037) dm-8 IBM,2810XIV
```

multipath -ll (detailed)

```
## This example shows my connections to a IBM XIV SAN, yours may look different

server1> multipath -ll

mpath2 (20017380006c00036) dm-7 IBM,2810XIV
[size=96G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=1][active]
\_ 0:0:2:3  sdai 66:32   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:3:3  sday 67:32   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:4:3  sdbo 68:32   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:5:3  sdce 69:32   [active][ready]
```

```
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:0:3  sdcu 70:32   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:0:3  sdc  8:32    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:1:3  sddk 71:32   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:2:3  sdea 128:32  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:3:3  sdeq 129:32  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:4:3  sdfg 130:32  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:5:3  sdfw 131:32  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:1:3  sds  65:32   [active][ready]

mpath1 (20017380006c00035) dm-6 IBM,2810XIV
[size=32G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=1][active]
\_ 0:0:2:2  sdah 66:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:3:2  sdax 67:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:4:2  sdbn 68:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:0:2  sdb  8:16    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:5:2  sdcd 69:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:0:2  sdct 70:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:1:2  sddj 71:16   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:2:2  sddz 128:16  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:3:2  sdep 129:16  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:4:2  sdff 130:16  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:5:2  sdfv 131:16  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:1:2  sdr  65:16   [active][ready]

mpath0 (20017380006c00034) dm-5 IBM,2810XIV
[size=32G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=1][active]
\_ 0:0:2:1  sdag 66:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:3:1  sdaw 67:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:0:1  sda  8:0     [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:4:1  sdbm 68:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:5:1  sdcc 69:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:0:1  sdcs 70:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:1:1  sddi 71:0    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:2:1  sddy 128:0   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:3:1  sdeo 129:0   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:4:1  sdfe 130:0   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:5:1  sdfu 131:0   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:1:1  sdq  65:0    [active][ready]

mpath3 (20017380006c00037) dm-8 IBM,2810XIV
[size=368G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=1][active]
\_ 0:0:2:4  sdaj 66:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:3:4  sdaz 67:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:4:4  sdbp 68:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:5:4  sdcf 69:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:0:4  sdcv 70:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:1:4  sddl 71:48   [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:0:4  sdd  8:48    [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:2:4  sdeb 128:48  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:3:4  sder 129:48  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:4:4  sdfh 130:48  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 1:0:5:4  sdfx 131:48  [active][ready]
\_ round-robin 0 [prio=1][enabled]
\_ 0:0:1:4  sdt  65:48   [active][ready]
```

If you have made a mistake in the **multipath.conf** file use the below to correct it

1. edit the **/etc/multipath.conf**
2. service multipathd reload
3. multipath -F
4. multipath -d (again double check that you are happy)
5. multipath -v2

On occasion you may come across where the array you have is not in the **multipath.conf.defaults** file, you can add a devices section, you may need to consult the manufactures documentation on what you may need in the **multipath.conf** file, below is an example of a HP OPEN-V series array.

non-default array support	```## add to the multipath.conf file
	
	devices {
	   device {
	            vendor "HP"
	            product "OPEN-V."
	            getuid_callout "/sbin/scsi_id -g -u -p0x80 -s /block/%n"
	   }
	}
	
	Note: make sure you consult the release notes regarding your array```

Configuration file in detail

The configuration file is divided into the following sections

- blacklist - listing specific devices to exclude from multipathing
- blacklist_exceptions - listing of multipath candidates that would otherwise be excluded (blacklisted)
- defaults - general setup parameters
- multipaths - settings for the characteristics of individual multipath devices
- devices - settings for non-default storage arrays

You can blacklist any device you may wish but you need to tell multipath what to exclude, some examples would be the following

wwid	```## specific wwid
	blacklist {
	      wwid "20017380006c00034"
	}
	
	## All wwid
	## specific wwid
	blacklist {
	      wwid "*"
	}```

device name	```## all sd devices a through z
	blacklist {
	      devnode "^sd[a-z]"
	}
	
	## A more advanced example
	blacklist {
	   devnode "^(ram|raw|loop|fd|md|dm-|sr|scd|st)[0-9]*"
	   devnode "^(hd|xvd|vd)[a-z]*"
	}```

device type	```## blacklist HP devices
	blacklist {
	   device {
	            vendor "HP"
	            product "*"
	   }
	}```

To exclude from the blacklist you create an exception list (blacklist_exception), its the same syntax as above

wwid	```## exclude a specific wwid
	blacklist_exceptions {
	      wwid "20017380006c00034"
	}
	
	## exclude all wwid
	## specific wwid
	blacklist_exceptions {
	      wwid "*"
	}```

device name	```## all sd devices x through z
	blacklist_exceptions {
	      devnode "^sd[x-z]"
	}```

device type	```## exclude HP devices
	blacklist_exceptions {
	   device {
	            vendor "HP"
	            product "*"
	   }
	}```

The default section has a number of parameters which can be changed

Parameter	Default Value	Description
udev_dir	/udev	Specifies the directory where *udev* device nodes are created.
verbosity	2	(RHEL 5.3 and later) Specifies the verbosity level of the command. It can be overridden by the -v command line option.
polling_interval	5	Specifies the interval between two path checks in seconds.
selector	round-robin 0	Specifies the default algorithm to use in determining what path to use for the next I/O operation.
path_grouping_policy	failover	Specifies the default path grouping policy to apply to unspecified multipaths. Possible values include: failover = 1 path per priority group multibus = all valid paths in 1 priority group group_by_serial = 1 priority group per detected serial number

		group_by_prio = 1 priority group per path priority value group_by_node_name = 1 priority group per target node name
getuid_callout	/sbin/scsi_id -g -u -s	Specifies the default program and arguments to call out to obtain a unique path identifier. An absolute path is required.
prio_callout		Specifies the the default program and arguments to call out to obtain a path weight. Weights are summed for each path group to determine the next path group to use in case of failue. "none" is a valid value.
path_checker	readsector0	Specifies the default method used to determine the state of the paths. Possible values include: readsector0, rdac, tur, cciss_tur, hp_tur (RHEL 5.5 and later), emc_clariion, hp_sw, and directio.
features		The extra features of multipath devices. The only existing feature is queue_if_no_path, which is the same as setting no_path_retry to queue.
rr_min_io	1000	Specifies the number of I/O requests to route to a path before switching to the next path in the current path group.
max_fds		(RHEL 5.2 and later) Sets the maximum number of open file descriptors for the multipathd process. In RHEL 5.3, this option allows a value of max, which sets the number of open file descriptors to the system maximum.
rr_weight	uniform	If set to priorities, then instead of sending rr_min_io requests to a path before calling selector to choose the next path, the number of requests to send is determined by rr_min_io times the path's priority, as determined by the prio_callout program. Currently, there are priority callouts only for devices that use the group_by_prio path grouping policy, which means that all the paths in a path group will always have the same priority. If set to uniform, all path weights are equal.
failback	manual	Specifies path group failback. A value of 0 or immediate specifies that as soon as there is a path group with a higher priority than the current path group the system switches to that path group. A numeric value greater than zero specifies deferred failback, expressed in seconds. A value of manual specifies that failback can happen only with operator intervention.
no_path_retry	null	A numeric value for this attribute specifies the number of times the system should attempt to use a failed path before disabling queueing. A value of fail indicates immediate failure, without queuing. A value of queue indicates that queuing should not stop until the path is fixed.
flush_on_last_del	no	(RHEL 5.3 and later) If set to yes, the multipathd daemon will disable queueing when the last path to a device has been deleted.
queue_without_daemon	yes	(RHEL 5.3 and later) If set to no, the multipathd daemon will disable queueing for all devices when it is shut down.
user_friendly_names	no	If set to yes, specifies that the system should using the bindings file to assign a persistent and unique alias to the multipath, in the form of mpathn. The default location of the bindings file is /var/lib/multipath/bindings, but this can be changed with the bindings_file option. If set to no, specifies that the system should use use the WWID as the alias for the multipath. In either case, what is specified here will be overriden by any device-specific aliases you specify in the multipaths section of the configuration file.
bindings_file	/var/lib/multipath/bindings	(RHEL 5.2 and later) The location of the bindings file that is used with the user_friend_names option.
mode	The default value is determined by the process.	(RHEL 5.3 and later) The mode to use for the multipath device nodes, in octal.
uid	The default value is determined by the process.	(RHEL 5.3 and later) The user ID to use for the multipath device nodes. You must use the numeric user ID.
gid	The default value is determined by the process.	(RHEL 5.3 and later) The group ID to use for the multipath device nodes. You must use the numeric group ID.
checker_timeout	The default value is taken from sys/block/sdx/device/timeout.	(RHEL 5.5 and later) The timeout value to use for path checkers that issue SCSI commands with an explicit timeout, in seconds.

The multipaths section parameters are as follows

Parameter	Description
wwid	Specifies the WWID of the multipath device to which the multipath attributes apply.
alias	Specifies the symbolic name for the multipath device to which the multipath attributes apply.
path_group_policy **prio_callout** **path_selector** **failback** **rr_weight** **no_path_retry** **flush_on_last_del** **rr_min_io** **mode** **uid_gid**	All the same as the defaults table

The devices section parameters are as follows

Parameter	Description
vendor	Specifies the vendor name of the storage device to which the device attributes apply, for example COMPAQ.
product	Specifies the product name of the storage device to which the device attributes apply, for example HSV110 (C)COMPAQ.
path_checker	Specifies the default method used to determine the state of the paths. Possible values include readsector0, rdac, tur, cciss_tur, hp_tur, emc_clariion, hp_sw, and directio.
features	The extra features of multipath devices. The only existing feature is queue_if_no_path, which is the same as setting no_path_retry to queue.
hardware_handler	Specifies a module that will be used to perform hardware specific actions when switching path groups or handling I/O errors. Possible values include 0, 1 emc, and 1 rdac. The default value is 0.
product_backlist	Specifies a regular expression used to blacklist devices by product.
path_group_policy **getuid_callout** **prio_callout** **path_selector** **failback** **rr_weight** **no_path_retry**	All the same as defaults table

flush_on_last_del rr_min_io	

Administration and Troubleshooting

Once you have multipathing configured and running there are a number of administering tasks that you may need to perform or use

- Resizing an online multipath device
- Issues with very large numbers of LUNs
- Issues with queue_if_no_path feature
- Multipath command output
- Multipath queries with multipath command
- Multipath command options
- Multipath queries with dmsetup command
- Troubleshooting with the multipathd interactive console

To resize a online multipath device you must follow the procedure below

Resize online multpath LUN		
1	Resize the physical device	`## use whatever method your SAN uses to resize the physical device`
2	find the paths of the devices	`multipath -l`

3	Resize the paths	`## For SCSI devices writing a 1 (one) to the rescan file for the device causes the SCSI driver to rescan`
		
		`echo 1 > /sys/block/<device_name>/device/rescan`

4	Resize the multipath device	`multipathd -k'resize map mpath0'`

5	resize the filesystem	`## use the method for the appropriate filesystem type`
		
		`# I am assuming that no LVM or DOS partition is being used`
		`resize2fs /dev/mapper/mpath0`

When a large number of LUNs are being used multipathed devices can increase the time it takes for the udev device manager to create devices nodes for them. You can correct this problem by "deleting/commenting out" the below line from the /etc/udev/rules.d/40-multipath.rules file. This line causes the udev device manager to run multipath every time a block device is added to the node. Even with this line removed, the multipathd daemon will still automatically create multipathed devices, and multipath will still be called during the boot process for nodes with multipathed root file systems. The only change is that multipathed devices will not be automatically created when the multipathd daemon is not running, which should not be a problem for the vast majority of multipath users.

/etc/udev/40-multipath_rules	`KERNEL!="dm-[0-9]*", ACTION=="add", PROGRAM=="/bin/bash -c '/sbin/lsmod | /bin/grep ^dm_multipath'", RUN+="/sbin/multipath -v0 %M:%m"`

If features "1 queue_if_no_path" is specified in the /etc/multipath.conf file, then any process that issues I/O will hang until one or more paths are restored. To avoid this, set the no_path_retry N parameter in the /etc/multipath.conf file (where N is the number of times the system should retry a path).

If you need to use the features "1 queue_if_no_path" option and you experience the issue noted here, use the dmsetup command to edit the policy at runtime for a particular LUN (that is, for which all the paths are unavailable). For example, if you want to change the policy on the multipath device mpath2 from "queue_if_no_path" to "fail_if_no_path", execute the following command.

change policy on multipath device	`dmsetup message mpath2 0 "fail_if_no_path"`
	
	`Note: you must specify the mpathn alias rather than the path`

The multipath command outputs data in the format as below

multipath device	`action_if_any: alias (wwid_if_different_from_alias) [size][features][hardware_handler]`
path group	`\_ scheduling_policy [path_group_priority_if_known] [path_group_status_if_known]`
path	`\_ host:channel:id:lun devnode major:minor [path_status] [dm_status_if_known]`

The *multipath* command has a number of options, which vary the detail of the output and can remove devices

multipath topology gathered from information in sysfs and the device mapper	`multipath -l`
displays as above plus in addition to all other available components	`multipath -ll`
no output	`multipath -v0`
outputs the created or updated multipath names only	`multipath -v1`
prints all detected paths, multipath and maps	`multipath -v2`
Remove the named device from multipath	`multipath -f <device>`
Remove all multipath devices	`multipath -F`
dry run, don't actually change anything	`multipath -d`

force maps to specified policy	`multipath -p <policy>`
	
	`Note: policies are`
	`failover`
	`multibus`
	`group_by_serial`
	`group_by_prio`
	`group_by_node_name`

You can also obtain details of the multipath*n* devices by using the *dmsetup* command

dmsetup	## List major, minor numbers dmsetup ls

The last command you can use is the *multipathd* command

multipathd	## Start interactive session multipathd -k ## Here is the help page multipathd> help multipath-tools v0.4.7 (03/12, 2006) CLI commands reference: list|show paths list|show maps|multipaths list|show maps|multipaths status list|show maps|multipaths stats list|show maps|multipaths topology list|show topology list|show map|multipath $map topology list|show config list|show blacklist list|show devices add path $path remove|del path $path add map|multipath $map remove|del map|multipath $map switch|switchgroup map|multipath $map group $group reconfigure suspend map|multipath $map resume map|multipath $map reinstate path $path fail path $path disablequeueing map|multipath $map restorequeueing map|multipath $map disablequeueing maps|multipaths restorequeueing maps|multipaths resize map|multipath $map

Multipathing and DR

I had a configuration where a server would be required to connect to two SAN's (one LIVE and one DR), the DR SAN was a mirror copy of the LIVE SAN. Both sets of SAN LUNs had different WWIDs, thus the configuration had to be changed when DR was invoked.

Disk	LIVE SAN wwid	DR SAN wwid
root	20017380006be0008	20017380006c00003
swap	20017380006be0009	20017380006c00004

The problem I had was when we cutover to the DR SAN the wwid did not match the existing configuration, and thus the multipathing was not working correctly, in order to get it working correctly three files must be changed

- /etc/multipath.conf
- /var/lib/multipath/bindings (if not using aliases in the multipath.conf file)
- /boot/initrd file

Firstly backup all the above files to a safe place as they will be needed when you cut back over to the LIVE environment.

/etc/multipath.conf	Change the /etc/multipath.conf to reflect the new DR wwid numbers
/var/lib/bindings	Then edit the /var/lib/bindings file to reflect the new DR wwid numbers, if you used aliasing in the multipath.conf then this file may not exist.
/boot/initrd file	The initrd (initialize ram disk) also has both the above files buried into it, thus we must unzip and un-cpio the image file make the changes and then re-cpio and zip it back again, the steps are below 1. make sure you have backed up the /boot/initrd image file 2. copy the /boot/initrd to a new directory (ready to unzip and uncpio) 3. rename the file so that it has a .gz at the end 4. gunzip the initrd file (gunzip initrd.img.gz) 5. uncpio the image file (cpio -id < initrd .img) 6. hopefully you should have a directory with bin, dev, etc, init, modules, proc,sbin, selinux, sys, tmp, var 7. remove the image file (we do not want to add this back into to the new image) 8. make the changes to the etc/multipath.conf and var/lib/multipath/bindings files to reflect the new DR wwids 9. now cpio the directory structure (find . | cpio --quiet -H newc -o | gzip -9 -n > /tmp/initrd_new.img 10. check the size difference with the original they should be of a similar size 11. now just copy over the existing initrd image file with the new one 12. reboot the server When the server reboots the new wwid should be in place and the server should use the correct multipathing for the DR SAN.

When you cut back to the LIVE server just copy the original files back and reboot the server.