Hi!

If you downloaded this firmware file use "Rufus" on a Windows machine (or dd on Linux, FreeBSD or similar) to write it (after unzipping it to an "img" file) *raw* to whatever your machine will boot. Most reasonably-modern computers will boot off a USB stick and some will boot from an SD card. *Note that the media is mounted read-only in normal operation and as such the system is* ***entirely power-fail safe****; you must explicitly use "save_cfg" to write changes you make to the running system or they will disappear at next power off or reboot.* This means you can test things provided you have not saved the configuration and, if it goes bad, simply pull the cord and plug it back in.

Note that it is built for and expects the drive the OS identifies this load on to be "da0"; most machines will be for a single USB storage device connected, so this image is suitable and built for a USB drive key. *Use a USB 3.0 key for best results;* older and slower sticks will work but you probably will not like the performance. If your machine identifies the device as something else (and you do not have other storage connected to USB) that can be fixed but if the cause is other connected storage devices *unplug them* since USB does not guarantee the order of connected things. Any stick >= 16Gb is fine.

Be aware that the update scripts in the root directory *may not do the right thing with setting the boot partition under EFI.* Specifically the EFI BIOS firmware may have a very different idea of disk orders than the system does once it boots, and there is no good way to detect this. Thus you will need to adjust to suit if you do "online" operating system/firmware updates.

The master configuration file for booting is in /etc/rc.conf. I've decently-commented it and it will need editing to suit your local requirements. You may need to adjust /usr/local/etc/dhcp6c.conf and /usr/local/etc/ipfw.local (the firewall config) and you WILL need to adjust /usr/local/etc/dhcpd.conf to suit. Specifically, the system comes configured to stick its own internal interface on 192.168.10.200, which you may not want, and it also is set up to have two subnets for dhcp clients (using VLAN capability), which you may want to change or not want at all. In addition the firewall script has to have both the internal and external interfaces defined ("iif" and "oif") which must match your hardware and how you have /etc/rc.conf (and the dhcp programs) set. If the VLANs are not desired while they are in the firewall configuration if the system isn't set up to use them that will not stop the firewall from working; it will simply complain about the lines that don't match a configured interface. As configured once you make sure the interface names are right the system will try to get a /60 prefix from your ISP for IPv6 along with a DHCP IPv4 address, and assign a /64 out of that (1 of 4) for local clients on the second Ethernet interface on which it will do SLACC (auto-configuration) for you. If your ISP refuses to hand out at least a /60 that will fail, but most (e.g. Cox, Medianet, etc) that currently allocate IPv6 will give you at least a /60, and some will give you a /56. **Many fiber providers ONLY hand out a /56; ask so you can match what the other end expects.**

You can **also** use "dhcpcd" *instead* of dhcp and dhcp6c to get both Ip4 and Ip6 addresses. Look in /usr/local/etc/dhcpcd.conf.sample for a reasonable configuration (you'll need to change interface names) and some explanation is in there, along with /etc/rc.conf. **Be aware that some ISPs absolutely insist on a consistent "duid" across boots and will misbehave badly if you don't -- and the factory (not**

dhcpcd) software *cannot do that* without a persistent place to store the generated value from the first boot. Dhcpcd has a way to do so and thus for many will be the preferred configuration. Of note some fiber providers are in this group.

The ipsec code (StrongSwan) is present but is NOT configured. Most of this is in /usr/local/etc, but not all – it's standard FreeBSD however, nothing fancy in that regard. Likewise, while snmpd is present the config file is not, and you will need to create it if you wish to use it for monitoring.

If you wish to save anything that's not on the ramdisk you need to mount the base directory read-write with "mount –o rw /", make your changes and then issue a "mount –o ro /". Beware that the second command may appear to "hang" for a fair bit of time – that's normal, do NOT pull the power while it's in that state or you're virtually guaranteed to scramble something. In normal operation the system is power-fail safe; it has nothing mounted writeable. There is a small, remaining-space partition on the card that can be mounted writeable and journaled if you wish but I recommend against it since that can scramble the card and cause the system to fail to boot on a power failure. Remember always that SD cards are designed for things like digital cameras, not general computer storage. They're very safe being read – being written, especially given that it's flash, not so much.

There is a script called "save_cfg" in the root user's home directory that must be used to save any changes you make to the running system's configuration; it mounts the configuration directory (which is normally NOT mounted), copies any changed files back, and then unmounts it. If you don't run that then pulling the plug reverts you to the last-saved configuration (very useful if you're screwing around, haven't saved it, and lock yourself out!) For obvious reasons I strongly recommend setting a root password and changing the "freebsd" one immediately.

Be careful putting these cards in PCs or other similar devices – you will probably get a prompt to "fix" them, and if you say yes, well, what's on there will be gone! ☺

Enjoy.

- Karl