- `http` - Boolean flag, enabled if present.

  If specified enable HTTP(S) Boot. Disable PXE Boot unless the `pxe` flag is also present. If neither flag is present, both are enabled by default.

- `https` - Boolean flag, enabled if present.

  If enabled, allow only `https://` URIs for HTTP(S) Boot. Additionally has the same behaviour as the `http` flag.

- `ipv4` - Boolean flag, enabled if present.

  If specified enable IPv4 for PXE and HTTP(S) Boot. Disable IPV6 unless the `ipv6` flag is also present. If neither flag is present, both are enabled by default.

- `ipv6` - Boolean flag, enabled if present.

  If specified enable IPv6 for PXE and HTTP(S) Boot. Disable IPV4 unless the `ipv4` flag is also present. If neither flag is present, both are enabled by default.

- `pxe` - Boolean flag, enabled if present.

  If specified enable PXE Boot, and disable HTTP(S) Boot unless the `http` or `https` flags are present. If none of these flags are present, both PXE and HTTP(S) Boot are enabled by default.

- `static4:{MAC_ADDR}[\VLAN_ID][="{IP},{MASK},{GATEWAY}[,{DNS}]"]` - String value.

  Specify static IPv4 address for the network interface with the MAC address given by `MAC_ADDR`. `MAC_ADDR` must be specified as 12 consecutive hex digits, with no spaces, colons or hyphens separating digit pairs. In some advanced use-cases such as iSCSI, the MAC address length may be some other even number length of hex digits. The required MAC address can be found in the names of the boot options produced by this driver. Note that hyphens separating digit pairs must be removed, as compared to the format displayed in boot option names. It is also possible to specify a VLAN ID to use on the interface, by adding a backslash followed by a 4 digit hex representation of the VLAN ID following the MAC address. The VLAN ID will also be shown in the boot entry name, but note that it must be converted from decimal in the boot entry name to a 4 digit hex number in this option.

  Required elements in value are IP address in `IP`, network mask in `MASK` and gateway in `GATEWAY`. Optional is an additional space separated list of one or more DNS servers in `DNS`. `DNS` will be needed if the boot file URI includes a domain name rather than an IP address.

  `MAC_ADDR` is not optional.

  If value is omitted, then any static IP for this MAC address (and VLAN ID when present) will be deleted.

  - Example 1: `static4:112233445566="192.168.1.20,255.255.255.0,192.168.1.1,8.8.8.8 4.4.4.4"`.
  - Example 2: `static4:112233445566\0001="10.0.0.2,255.255.255.0,10.0.0.1"`.

  Note 1: This option is written to NVRAM and will remain present even if the option is removed from the driver `Arguments`, unless ~~NVRAM is cleared or~~ an alternative value is written or the value deleted, using this option.

  Note 2: This setting will normally cause a static IP to be assigned during pre-boot, even in vendor-provided network stacks. However, due to a quirk of the design of PXE and HTTP boot, any such static assignment will then be ignored and DHCP used instead, during network boot. The OpenCore network stack (specifically `HttpBootDxe.efi`) is unusual in that it will allow HTTP boot from a static IP address, as long as an HTTP boot URI has also been specified, using the `uri` option for this driver (or e.g. in the OVMF admin screens if using OVMF, or similar options where present in other firmeare). If HTTP boot from static IP is required, then any pre-existing vendor-specific version of `HttpBootDxe.efi` will need to be unloaded (see `UEFI Unload` option) and the OpenCore version used instead.

required.

13. `ReplaceTabWithSpace`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Some types of firmware do not print tab characters or everything that follows them, causing difficulties in using the UEFI Shell's builtin text editor to edit property lists and other documents. This option makes the console output spaces instead of tabs.

    Note: This option only applies to `System` renderer.

14. `Resolution`
    Type: `plist string`
    Failsafe: Empty (Maintain current screen resolution)
    Description: Sets console output screen resolution.

    - Set to `WxH@Bpp` (e.g. `1920x1080@32`) or `WxH` (e.g. `1920x1080`) formatted string to request custom resolution from GOP if available.
    - Set to `Max` to attempt using the largest available screen resolution. When set to `Max` all available resolutions will be listed in lines starting `OCC: Mode` in the debug log.

    On HiDPI screens `APPLE_VENDOR_VARIABLE_GUID` `UIScale` NVRAM variable may need to be set to `02` to enable HiDPI scaling in `Builtin` text renderer, FileVault 2 UEFI password interface, and boot screen logo. Refer to the Recommended Variables section for details.

    Note: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with `ProvideConsoleGop` set to `true`.

15. `SanitiseClearScreen`
    Type: `plist boolean`
    Failsafe: `false`
    Description: Some types of firmware reset screen resolutions to a failsafe value (such as `1024x768`) on the attempts to clear screen contents when large display (e.g. 2K or 4K) is used. This option attempts to apply a workaround.

    Note: This option only applies to the `System` renderer. On all known affected systems, `ConsoleMode` must be set to an empty string for this option to work.

16. `TextRenderer`
    Type: `plist string`
    Failsafe: `BuiltinGraphics`
    Description: Chooses renderer for text going through standard console output.

    Currently two renderers are supported: `Builtin` and `System`. The `System` renderer uses firmware services for text rendering, however with additional options provided to sanitize the output. The `Builtin` renderer bypasses firmware services and performs text rendering on its own. Each renderer supports a different set of options. It is recommended to use the `Builtin` renderer, as it supports HiDPI mode and uses full screen resolution.

    Each renderer provides its own `ConsoleControl` protocol (in the case of `SystemGeneric` only, this passes some operations through to the system `ConsoleControl` protocol, if one exists).

    Valid values of this option are combinations of the renderer to use and the `ConsoleControl` mode to set on the underlying system `ConsoleControl` protocol before starting. To control the initial mode of the provided `ConsoleControl` protocol once started, use the `InitialMode` option.

    - `BuiltinGraphics` — Switch to `Graphics` mode then use `Builtin` renderer with custom `ConsoleControl`.
    - `BuiltinText` — Switch to `Text` mode then use `Builtin` renderer with custom `ConsoleControl`.
    - `SystemGraphics` — Switch to `Graphics` mode then use `System` renderer with custom `ConsoleControl`.
    - `SystemText` — Switch to `Text` mode then use `System` renderer with custom `ConsoleControl`.
    - `SystemGeneric` — Use `System` renderer with custom a `ConsoleControl` protocol which passes its mode set and get operations through to system `ConsoleControl` when it exists.

    The use of `BuiltinGraphics` is straightforward. For most platforms, it is necessary to enable `ProvideConsoleGop` and set `Resolution` to `Max`. The `BuiltinText` variant is an alternative to `BuiltinGraphics` for some very old