Chrome OS

Copied from Rob Sickler

Chrome OS is Google's fork of the Chromium project which is based on the Linux kernel. Google's first release of the OS was in the summer of 2011. Many schools like to use Chromebooks, which run Chrome OS, because Google's ecosystem is pretty friendly and reliable. They are also marketed toward first-time computer users due to their simplicity. I personally have a few Chromebooks and like to tinker a bit with them.

Working with Chromebooks

Working on a Chromebook (CB) is pretty simple; there's really not much to them. In most modern CBs, there's a motherboard, a wireless NIC (WNIC), a battery and a daughterboard for an additional USB port. Some of the *older* systems were initially designed as netbooks so they may have a cooling fan.

Show Version & Serial

You can log into the device and check the version but it's nice to figure it out before going through all that. For instance, collecting a few hundred CBs and deprovisioning them in the Google Admin Console is easier when you can just see the serial without having to log in. Or, if you're in the process of upgrading the OS on a mess of them, you can see the version before you go through the process of upgrading one that didn't need said upgrade.

- To show the **version** and the **serial number**, hit **Alt + V** at the **login screen** or the **Welcome** after a powerwash.

In the Google Admin Console, you can also add other info. For instance, adding the **asset ID** in the admin console will show it under the serial and build info when you hit **Alt + V**.

Powerwash a Chromebook

One of the more common things I've seen done to CBs is a powerwash. Basically, the powerwash sequence allows you to revert it back to *factory defaults*. In doing so, all local profiles will be wiped. However, that's generally not a big deal as Google's ecosystem tends to save everything in the cloud anyway. Generally speaking, you can powerwash your device all you want. Once you log back into it, all your settings and content come back down from the cloud.

Managed Chromebooks

The sequence is pretty simple but is still a little different than an unmanaged device. Most official[1] Chromebooks have their *standard* keyboards - which are different than a keyboard you'd find on a typical laptop. The following key sequence can be used regardless whether or not the device is on and logged into. It uses the top row of keys...

- To start, you need to give the device the *three-finger salute*:
  - Hold down **Escape** and **Refresh**[2] and hit the **Power** button.
- If the device is on, it should reboot to a **white screen**, nagging about a **missing or damaged OS**. If the device is off, it should boot up to said screen.
- At the aforementioned screen, you hit **Ctrl** + **D**.
- At the next screen, hit **Enter** and it'll **turn off OS verification** and reboot.
- At the next screen, hit the **Space** bar to **re-enable OS verification**.
- At the next screen, hit **Enter** to **confirm** the re-enabling of OS verification.

After that, it should boot up and look like it's the first time anyone is logging into the device. That's basically true as there should be no more profiles on the device once it has been reverted back to the *factory default* state.

Unmanaged Chromebooks

You can use the steps, outlined above, to powerwash a CB but unmanaged devices typically have a couple ways of doing it. These additional ways are outlined below.

Through Settings

This can be used via the GUI while logged in. I've had to use this method when certain keys didn't work.

1. Head into **Settings** via the system tray.
2. If you've not already seeing **Reset Settings**, show it via the *hamburger* menu[3] in the left pane by selecting **Advanced**.
    1. You can also see this by scrolling down to the bottom of the page; it should be the last thing listed.
3. Under **Reset Settings**, you should see the section labeled, **Powerwash**.
    1. If your device is managed, you may not see this listed as it may have been disabled by your admin.
4. Follow the prompts to powerwash your device.

4-finger Salute

A quick keyboard combo will do the same thing - so long as it hasn't been disabled by your admin.

1. Ctrl + Alt + Shift + R
2. Follow the prompts.

Reboot a Chromebook

Sometimes, a powerwash isn't really needed; sometimes you just need a reboot. Powering off the device and bringing it back up isn't the same, from what I've read.

- Remember that **Refresh** key I mentioned earlier? Well, hold it down and hit the **Power** button.
- If the sequence is done successfully, and the keys are in working order, the device should immediately reboot.

Developer Mode

Developer Mode is a special mode that allows you access to lower areas of the OS and device. Chromebooks have several layers of access and Developer Mode allows you to do things you just can't do in normal mode. I typically get into it so I can install the RMA SHIM Tool from the manufacturer and/or change the serial number[4] while in a virtual console.

Managed Device:

If the device is managed via the **Google Administration Console**, you may not be able to enter Developer Mode as it may be blocked. In those cases, you will likely need to **deprovision** the device first. Moreover, you may need to deprovision it while it's in contact with the management servers - which can be tough when the device doesn't want to boot up and make said connection.

Write-Protection:

This is an **optional** step and you only need to do this when you actually want to **write changes** to the chip on the motherboard. There is no single answer as it will likely vary between manufacturers and models. Some models have a screw while others have a soft switch[5] The easiest thing to do is search the web for pictures and help for your specific device. Write-protection is in place so you can't write data to a certain chip on the board.

For those that have a screw, it's typically one of the screws holding in the motherboard. In my experience, the write-protect screw is typically threaded through the only hole in the motherboard that has **solder points/beads circling the hole** so, when the screw is fully seated, it completes or shorts a circuit and prevents write-access. Furthermore, the screw is typically covered by a small **sticker** - unless you've already been inside, messing around and removed it.

Enter Developer Mode

The sequence isn't overly difficult but there are several steps and some can be skipped - depending on what you're trying to accomplish.

- This first step is **optional**:
  - As noted above, find and remove the write-protect screw if you plan to save changes to an otherwise write-protected chip on the board. This generally involves taking the CB apart to some degree so be careful.
- Do the first few steps of the *three-finger salute* like one does when they're about to power wash a CB: **Esc** + **Refresh** + **Power**
- When you get to a white screen, nagging about a missing or damaged OS hit **Ctrl** + **D**.
- On the next screen, hit **enter** and it should reboot.
- When it comes back up and shows you a screen nagging about the **OS verification** being **turned off**, you can either wait for a few minutes or you can skip the wait and hit **Ctrl** + **D** again.
  - You will see this screen every time you boot up so just wait for it to continue to boot.
- It should reboot again and begin heading into Developer Mode. This process takes a while so walk away and find something else to do for 20 minutes.
  - If the device is (or still thinks it is) **managed**, this is the point it will **deny** access to Developer Mode. It will then take a few more minutes and reboot again but you'll go back into *normal* mode.
- If it's successful, it'll reboot again, *pause* for a bit at the screen nagging about **verification** being **turned off** and it'll begin to boot into the OS after a few seconds.

Exit Developer Mode

There are a couple ways you can use to get out of Developer Mode...

If you plan on enrolling the device, be sure to **exit developer mode prior to enrolling**. If dev-mode is not allowed by the organization, you may end up in a **boot loop** which may nag about how you're not allowed to be in dev-mode and reboot. The only way I've found to get out of this loop is to disable write-protection, install the RMA SHIM Tool and start over.

Typical Way to Exit Developer Mode

- You can exit Developer Mode via a few keystrokes:
  - To exit Developer Mode, you merely need boot up and hit the **space bar** at the screen nagging about **verification** being **turned off** - which is seen every time the device boots up. Again, it'll take a hot minute and then it'll reboot and go back to *normal* mode. Just remember to put the write-protect screw back in - if you removed it before.

Advanced Way to Exit Developer Mode

You can also exit Developer Mode via the command line - which may also help when you **can't** seem to get out of Developer Mode. Sometimes, when you attempt to get out of Developer Mode, you'll see some black & white text in the upper left corner of the screen. In said text, you may see a message like:

```
WARNING: TONORM prohibited by GBB FORCE_DEV_SWITCH_ON
```

When you see the aforementioned warning, you'll likely have to use the method below in order to get out of Developer Mode:

- Inside a superuser session, like what you'd use to change the serial number, run something like this:

  ```
  /usr/share/vboot/bin/set_gbb_flags.sh 0x0
  ```

  - If you're not in a superuser session, you can prefix the command with `sudo`.
- Reboot the device with the following command when the aforementioned command completes:

  ```
  reboot
  ```

- After the reboot, it should transition out of Developer Mode. However, now and then, it'll nag about missing its OS. No worries; you can reload it with a USB thumb drive or an SD card via the Chromebook Recovery Utility.
- Don't forget to reinstall the **write-protect screw**.

Changing VPD Info

Changing the Serial Number

You typically don't need to change the serial number unless you've changed the motherboard and wish to keep the serial in sync with what's listed on the bottom of the device. I've replaced several for devices that are under warranty. Yes, I could have just sent the device back to the manufacturer but where's the fun in that?

Serial numbers are *burned* into the motherboard so you need to remove the **write-protect screw** if you want to change the serial number.

For some older devices, when I swapped out their motherboard and changed their serial number to match that which was noted on the bottom of the device, the WNIC got **very, very <color red>hot</color>** and the device could never connect to a wireless network. Moreover, the WNIC would get so hot that the screw, holding it in place, would heat up enough to damage the plastic case! Luckily, the WNIC gets noticeably overheated

within a few minutes of changing the serial number and powering on the device so you'll be able to shut it down and pull the WNIC pretty quickly so long as you don't button everything up too quickly. For the devices that were still under warranty, I would just order another motherboard & WNIC and go through the process again - until I found a combination that no longer cooked the WNIC. Just keep this in mind and only change the serial number when you absolutely need to.

To get around the issue with the **overheating WNIC**, I tend to set the WNIC aside when I'm rebuilding the CB. The WNIC is not necessary for changing the serial so I just leave it out of the equation. I go through the process of changing the serial and, before I button everything up, I install the WNIC. Now and then, I'll still have one that fails to be *seen* by the device. Luckily, I have several on hand so I just swap them out until I find one that works. Typically, once you find one that works, you can swap the original back into place and it'll *magically* work again.

Changing the Serial via the Virtual Terminal

You should easily be able to do this with the RMA SHIM Tool from the manufacturer but I've had mixed results with that. On some occasions, it worked. Most, however, have failed miserably. Because of the failure rate, I tend to use a virtual terminal session. The virtual terminal sessions allows for some low-level access to the OS at a command line. Most Linux distros have several virtual terminal sessions you can access.

- You need to be in Developer Mode before you can get to the virtual terminal session so do that and then come back here to continue.
- Once you're in Developer Mode, you can hit **Ctrl** + **Alt** + **→**[6] once it's booted into the OS.
- Within a second or so, you should be dropped to a **command prompt**.
- Once at the command prompt, you are prompted for a **username** so enter:

```
chronos
```

- Then, you need to work as a super user so enter:

```
sudo su
```

- You'll want to check the currently programmed serial number, according to the Vital Product Data, so enter:

```
vpd -l
```

    - You may also see other things mixed in with the info but you're looking for something like `mlb_serial_number` and `serial_number` as those will be what you want to change Just find and replace the appropriate field(s). In some cases, I've changed both. Even when there was only one to begin with, doing the other will only add it and I've not had any issues with making sure both are in place when only one was there initially. In fact, I've had instances

where having the `mlb_serial_number` but **not** the `serial_number` kept the device from enrolling back into our management.
    - You may also see a **warning**, stating something like the following but, the warning seems to go away after you add/change the serial number:

```
[warn] vpd partition not formatted
```

- To change the values, enter something like the following but make sure any **letters** are **capitalized**:
    - serial_number:

```
vpd -s "serial_number"="99999999999999999999"
```

    - mlb_serial_number:

```
vpd -s "mlb_serial_number"="99999999999999999999"
```

        - The `mlb_serial_number` entry is typically found to be something other than the serial number when you dive into machines fresh out of the box. However, I've been making the same as the serial number without any issues when doing my repairs.
- Might as well verify the serial number again:

```
vpd -l
```

- If you've **messed up**, like I've done with a random capital letter in the entry, you can remove the entry with a command like:

```
vpd -d "serial_Number"
```

- You can make other VPD tweaks if you'd like.
- Once you're satisfied, *flush* the logs:

```
dump_vpd_log --force
```

- If you need to note the MAC address of a built-in WNIC for your radius server or MAC address white-list, you might as well do that before you reboot.
- Once you've changed the serial, you can enter the following command or hold down the **refresh**[7] button and hit the **power** button to reboot:

```
reboot
```

- When it reboots and comes back to the screen, nagging about **verification** being **turned off**, you can hit the **space bar** to enable OS verification and then hit **enter** to reboot.
    - If this gives you trouble, you may have to do use a more advanced method of getting out of Dev-mode.
- Don't forget to revert whatever change you made to disable **write-protection**.
- Because of the aforementioned warning, I tend to leave the case open until I get the device connected to a wireless network. If the WNIC works and doesn't overheat within the first couple of minutes, I'll close the case and secure all the screws.

Changing Other VPD Entries

The section above that describes how one can change the serial number via some VPD commands but you can also tweak some other things. This comes in handy for anyone who wants to use the Guest Login feature. It sets the aforementioned settings at the login screen. So, if you were an American living in Canada with a Chromebook with Canadian firmware, and you had American friends who would come to visit, you could set these tweaks and the users wouldn't get Canadian settings at the login screen. You'll see the settings I've used below but you can find more here if you wanted other locales and timezones.

- Change the locale:

```
vpd -s "initial_locale"="en-US"
```

- Change the timezone:

```
vpd -s "initial_timezone"="America/New_York"
```

- Change the keyboard layout:

```
vpd -s "keyboard_layout"="xkb:us::eng"
```

- Change the region:

```
vpd -s "region"="us"
```

- Specify the model:

```
vpd -s "model_name"="Lenovo 300e Chromebook 2nd Gen MTK"
```

- Specify the asset tag:

```
vpd -s "asset_tag"="IT-12345"
```

- Specify the machine type:

```
vpd -s "mtm"="81QC"
```

- Specify the Service Tag[8)]:

```
"service_tag"="1GWZ083"
```

- Specify the manufacturer date:

```
"mfg_date"="2020-11-28"
```

- Specify the WNIC's MAC address:

```
"wifi_mac0"="04:6c:59:31:1b:ff"
```

Changing the Hardware ID

I've had several instances where a bad HWID kept me from getting updates. All I could do was reload the OS via USB to get the device updated; I couldn't update it via **Settings > About Chrome OS**. After that, you were still *stuck* at that version until you did another manual update.

For us, it was largely an issue with Lenovo 100e (81ER) motherboards that had been swapped out while they were under warranty. They were coming to us with a *test* HWID and that wouldn't allow us to update the device via typical means. We didn't notice it because our SOP is to reload the OS via USB since we don't have very good luck with the SHIM tool. Naturally, the OS on the installation media is up-to-date so we never knew they weren't going to update after they left our workbench. In retrospect, we should have been doing regular checks regarding the versions of Chrome OS on our network.

The SHIM tool typically handles this but we have bad luck with it and it rarely finishes without errors. So, try your SHIM tool if you have one that works. If applicable, the hardware-level write protection may need to be disabled for this to work. The device **needs** to be in Developer Mode so make sure you have those rights before you proceed.

Our CBs are **managed** via Google Admin Console (GAC) so the process below may differ from your process. However, the basic steps involving the command line should still be the same.

Manual Process

- Log into the affected (and enrolled) CB with any account.
- Go to the following URL once you're logged in: chrome://policy/
- Log into GAC and **deprovision** the device.
- Back on affected CB, with the **policy** page already open, use the provided button to **reload** the policy.
    - You should see the **Status** change on the policy page, showing that it has been deprovisioned.
- **Reload** the **OS** via USB or SD card.
    - Sometimes you can get around this step with a couple of power wash sequences.
- Log in with a user that's not *locked down*.
    - Your personal account will suffice. Just make sure the user has the rights to be an **owner** of a device. Student accounts typically do not have said rights.
- Do what is needed to get into Dev Mode.
- Once you're in Dev Mode, hit **Ctrl** + **Alt** + **→**[9] once it's booted into the OS and sitting at the typical welcome screen.
- You should be dropped into the shell, asking for creds.
    - User: **root**
    - Password: You should not be prompted for a password…

- Once logged in, run the following commands[10]:
  - You must be in the **/tmp** directory; it won't work in your home directory:

```
cd /tmp
```

  - **Read** the **GBB** section of the flash chip and dump the data to a **BIN** file:

```
flashrom --read --image GBB:gbb.bin
```

    - If you'd like to see the HWID being reported, you can get it out of the file you just created:

```
gbb_utility --get gbb.bin --hwid
```

  - **Write** the **HWID** to the **BIN** file:

```
gbb_utility --set --hwid "ROBO D5B-B4K-E5Q-45M-Y8C-A92" gbb.bin
```

    - The aforementioned HWID is a valid one for Lenovo 100e Chromebooks, type **81ER**.
      - If you need another HWID, maybe this page will help.
    - The **HWID** is always **capitalized**.
  - **Flash** the GBB section of the device with the **updated** BIN file:

```
flashrom --write --image GBB:gbb.bin
```

- **Reboot** the device via the following command:

```
reboot
```

- Log in with the guest account and **test** whether or not you can **update** the OS via **Settings > About Chrome OS**.
- Reboot the device and exit Dev Mode
  - You should just be able to hit the **space bar** to get out of Dev Mode.
  - Re-enroll the device.

Useful Terminal Commands

Within this documentation, there are several examples of commands being used in the virtual terminal. Here are some more.

Finding the MAC Address

This comes in handy when you're replacing a motherboard with a built-in[11] WNIC and you run a radius server in your wireless environment. You're already in the terminal, *reserializing* the board so you might as well grab this bit of info if you need it for your radius server or MAC white-list.

- Get the MAC address:

```
connectivity show devices | grep -i "address:"
```

  - In most cases, you'll only have one address because you'll only have one adapter preset so your output should look similar to:

```
Address: a81d16157bd7
```

Changing the Date

Now and then, you need to use the RMA SHIM Tool to update firmware and whatnot for these things. Ideally, one should use it to prep a CB for the end user but it will **often fail** to *Finalize* for various reasons. One reason is the failure to **write the HWID**. Sometimes you can get beyond this error by **manually setting the date**.

- Set the date in the virtual terminal[12]:

```
date -s "20191231"
```

System Diagnostics

In Chrome OS, you can get into a System Diagnostics page. It allows you to see certain things that aren't found in the typical Settings screens. You can see the OS version number, hardware class, MAC address, IP address, memory usage, the battery's model number, various logs, etc.

- To see this page, open the browser and enter the following address: **chrome://system**
  - Find the **MAC address** under the `ifconfig` and `network-devices` sections.
  - Find the **IP address** under the `network-devices` section.

Battery Disconnect

No, this is not about taking apart the device and unplugging your battery... This is how CBs are shipped. There's a sequence that allows the device to be powered down and a *software switch* disconnects the battery for long-term storage. To get out of this mode, you have to plug the CB into a power source - just like you have to do when you unbox a new CB.

- Charge the device.
  - Ideally, a full charge is nice but Google says 80% is fine.
- With the device plugged into a power source, make sure the device is powered on.
- While holding the **refresh button**[13] and the **power button**, pull out the charging cable.
  - With touchscreens like Lenovo's 300e CBs, the power button is on the side. It may be difficult

to reach both the buttons with one hand so I tend to use two hands for the buttons while the device is in my lap. I then use my foot to step on the power cord and then lift the CB away - popping out the power cord.

To get the device to power on after being put into a battery-disconnect state, simply plug the device in and power it on.

[1)]

Not a laptop loaded with Chromium or Neverware's CloudReady.

[2)]

The **Refresh** key is generally referred to as the **F3** key if you're ever poking around forums.

[3)]

three lines

[4)]

Serial numbers in CBs are typically *burned* into the board.

[5)]

With a Lenovo 100e, you have to unplug the battery to disabled write-protection.

[6)], [9)]

The → key is typically referred to as F2.

[7)]

typically referred to as F3

[8)]

typically found on Dell CBs and is typically set the same as the serial

[10)]

the commands are **case-sensitive**

[11)]

soldered to the mobo

[12)]

Format: YYYYMMDD

[13)]

F3