OSKR Owner's Manual

Digital Dream Labs

Jul 20, 2021

CONTENTS:

1 Intro **1**
1.1 Welcome . . . 1
1.2 Copyright . . . 1
1.3 Distribution Notes . . . 2

2 Unlocking your Robot **3**
2.1 What is Unlocking? Why do we do it? . . . 3

3 Checklist **5**
3.1 Preparation 1: Getting your QSN and submitting it to Digital Dream Labs . . . 5
3.2 Peparation 2: Setting up your computer to upgrade Vector . . . 5
3.3 Preparation 3: Prepare Vector for the upgrade . . . 6
3.4 Deploy 1: Test Deploying the latest Production Software OTA image . . . 6
3.5 Deploy 2: Deploying the Unlock OTA image . . . 7
3.6 Deploy 3: Deploying the latest OSKR Software OTA image . . . 7
3.7 Getting Secure Shell . . . 8
3.8 If you need to turn Vector into an Access Point to get the Key . . . 8

4 Detailed Unlock Steps **9**
4.1 Step 1: Getting your QSN . . . 9
4.2 Step 2: Doing a Test Upgrade . . . 9
4.3 Step 3: Installing the unlock image . . . 14
4.4 Step 4: Installing the latest OSKR image . . . 14
4.5 Sign in to Vector . . . 15
4.6 Step 5: Getting your ssh key . . . 15
4.7 Congratulations! Vector is unlocked! . . . 16
4.8 Advanced Network Configuration: Enabling AP mode . . . 16

5 Tools **19**
5.1 Vector Self Test . . . 19
5.2 SSH / Shell Access . . . 19
5.3 Compilers . . . 20
5.4 Text editors . . . 20
5.5 logcat . . . 20
5.6 Development Web Servers . . . 21

6 Filesystem layout **25**
6.1 system partition . . . 25
6.2 /anki directory . . . 25
6.3 /data partition . . . 25
6.4 Editing the system partition . . . 26

7 Vector subsystems **27**
7.1 vic-switchboard 27
7.2 vic-cloud 27
7.3 vic-anim 27
7.4 vic-gateway 27
7.5 vic-engine 27
7.6 vic-dasmgr 28

8 Security **29**
8.1 BLE 29
8.2 Securing SSH 29
8.3 Unsecuring Development Tools 30
8.4 Manifest signing keys 31

9 Example Customization Tasks **33**
9.1 Disable Automatic updates 33
9.2 Change Boot Animation 34
9.3 Edit Phrases 34
9.4 Customize Eye Color(s) 35
9.5 Watch Face Recognition via Console Variables 36
9.6 Simulate a crash with a console function 37

10 Open Source Repository List **39**
10.1 Software Development Kits 39
10.2 Animations 39
10.3 Audio 39
10.4 Cloud / Hosted Service Communications 39
10.5 Escape Pod 40
10.6 Utilities 40
10.7 Documentation 40

11 Appendix 1: Vector Error Codes **41**
11.1 Ota Error Codes 41
11.2 Runtime Error Codes 42

12 Appendix 2: Vector Maintenance Tasks **43**
12.1 Factory Reset 43
12.2 Erasing User Data 43
12.3 CCIS Screen 44
12.4 Backing Up JDOCS 44

13 Appendix 3: Living with Windows **45**
13.1 PuTTY - A good ssh shell for Windows 45
13.2 Full Linux Environment 45

14 Appendix 4: Resources **47**

15 Glossary **49**

CHAPTER

ONE

INTRO

1.1 Welcome

The **Open Source Kit for Robotics** aims to provide real low level access to a consumer grade robot in a way that's completely unprecedented. In the past if you wanted to program a robot you've had to choose between:

- Polished high-quality consumer grade electronics that may or may not provide limited exposure and APIs to program at the most superficial level.
- Home-brew projects running on hobbyist level hardware with simple programs of a few hundred lines of code to create interesting but simplistic behaviors.

Our goal at Digital Dream Labs is to eliminate these trade-offs and provide you with direct access to:

- The internals of a robot with all the power of a smart phone.
- A state-of-the-art industrial strength codebase with hundreds of thousands lines of code.
- Sophisticated tools to debug and analyze code performance.
- Packaging tools to distribute your custom versions of the software to other robotics enthusiasts.

Please note that there will be some small changes due to contractual or legal requirements. Review these changes **before** proceeding:

- It will not be possible to go back to a production robot. The unlock process is permanent.
- Amazon Alexa mode is disabled. Due to contractual requirements we are unable to include Alexa on Vectors where users can modify the system without review by Amazon.

We are excited as you are for this opportunity. We look forward to seeing all the ways people will transform Vector and find completely new uses that we couldn't even conceive or imagine. We feel no matter how high we set our expectations someone out there will find ways to surprise the world.

1.2 Copyright

1.3 Distribution Notes

This version of was written for OSKR software version 1.7.1.

The latest official version of this document can be found on the web at https://oskr.ddl.io/oom/. A pdf version is also available at https://oskr.ddl.io/oom/oskrownersmanual.pdf.

This document was generated by the Sphinx Documentation Generator and the original source is available at https://github.com/digital-dream-labs/oskr-owners-manual.

CHAPTER

TWO

UNLOCKING YOUR ROBOT

The first step in your new adventure is unlocking your robot and transforming it from a normal Vector to and OSKR enabled Vector where you will be able to directly access all internals.

2.1 What is Unlocking? Why do we do it?

By default the operating system of Vector is secure and unchangeable with digital signatures to ensure system integrity. The security starts with encryption keys baked in to the hardware itself and works its way down to the final operating system image. In each phase of the boot process the previous system verifies the next system loads. If even one bit, byte, letter, etc change in any of the phases then the system will stop loading and come to a halt. This is done for:

- **Security** We want to make sure a malicious user hasn't modified the code to do something that it shouldn't be doing, like sending spam emails. Even if a hacker or bot network is somehow able to access the Vector they won't be able to make any changes to turn it in to the wrong kind of bot.
- **Reliability** We don't want mistakes to break the device. There are currently hundreds of thousands of Vectors out there, and we're hoping for millions and billions. At that level even small problems can add up quickly. Ensuring the Operating System hasn't been changed ensures reliability.

At boot time the following verification happens.

1. The hardware chip itself verifies the cryptographic signature on the low level ABOOT partition. It is for all practical purposes impossible to override this check.
2. The ABOOT partition has another security key baked in that it uses to verify the BOOT partition and load it. The BOOT partition is a skeleton linux operating system to bootstrap the main system.
3. The BOOT partition loads the appropriate SYSTEM partition which is the final software to run.

   In the case of a normal Vector the SYSTEM partition is protected by an Android-specific system called `dm-verity` to ensure that this partition isn't modified.

   Development Vectors used internally by the company, and now OSKR Vectors do not enforce `dm-verity` security on the system partition allowing development of new features.

In addition to these layers of cryptographic security a new Vector comes with a **recovery filesystem** installed for reliability purposes. This has been loaded on at the factory and is the boot system of last resort when no other software will load.

We then set aside two slots for day-to-day software usage called **A** and **B**. When you install an OTA update the system:

1. Downloads the update.
2. Performs some basic verification to see if the image 'fits' with the current configuration.
3. Decides if the A or B slot is the available one.

4. Copies the new filesystems on to the slot.
5. Flags the new slot as the ‘good’ one.
6. Reboots and loads the software. If the software fails to load it marks the slot as invalid.

   If there is a previously valid slot, such as **A** after an update to **B** was installed, it reverts to that.

   If there is now good **A** or **B** slot it reverts back to the **recovery filesystem** so you can try to install again. This is important. No matter how bad the running system gets messed up we should **always** be able to fall back to the **recovery filesystem**

So what does all this mean? We’ve got two conflicting goals:

1. Normal Vectors should have all the security and reliability that are expected for a consumer electronic device and need to remain locked down.
2. OSKR Vectors should have some ability to lock down the software installed but some ability to modify software for development.

Because of this each type of robot need to enforce different requirements each needs a different set of cryptographic signatures so that a normal Vector behaves the way it should and a OSKR unit behaves the way it should.

To accomplish this and unlock a OSKR robot we need to:

1. Reprogram the ABOOT image with a new cryptographic signature.
2. Sign the new BOOT images with this signature.
3. Create new **recovery filesystems** signed by the new keys because the old ones will be considered ‘bad’ when we swap out the signatures.

This is where things get tricky. The normal assumption is that the ABOOT and recovery filesystems get installed directly at the factory and physically written directly to the chips before final assembly of the robot, and never touched again.

But to do this outside of the factory we need to do this with our OTA (Over The Air) update scripts. And we need to modify software partitions that were never expected to be written in an environment much less reliable than a factory with physical connections to the hardware. So we must be very careful to make sure that we generate correct images, and the installation process completes without interruption.

But enough of the theory let’s dive in to the process…

CHAPTER

THREE

CHECKLIST

This is a detailed checklist of the steps to follow in unlocking your robot. I recommend printing it out for each robot you unlock, and ticking off each item, in order. The sections after this checklist will describe how to perform each step in more detail.

3.1 Preparation 1: Getting your QSN and submitting it to Digital Dream Labs

- [ ] 1. Open https://vector-setup.ddl.io in a Chrome browser
- [ ] 2. Place Vector in its charging station and press the backpack button twice to enter pairing mode.
- [ ] 3. Click "Pair with Vector" and select Vector from the popup.
- [ ] 4. Click the **Download logs** link
- [ ] 5. Find your QSN and ESN in those logs
- [ ] 6. Write down the QSN and ESN on this page – we'll double check this below.
- [ ] 7. Submit your QSN and ESN to Digital Dream Labs

3.2 Peparation 2: Setting up your computer to upgrade Vector

While you wait for your unlock OTA, you can set up your computer and prepare Vector. First, set up a HTTP server to deliver OTA files to your robot:

- [ ] 8. Install python on your computer, if you haven't already
- [ ] 9. Download the last production software from http://ota.global.anki-services.com/vic/prod/full/latest.ota
- [ ] 10. Start a simple webserver with the latest.ota
- [ ] 11. Find the IP address for your computer
- [ ] 12. Check that you can connect to this local webserver with Chrome.

3.3 Preparation 3: Prepare Vector for the upgrade

- [ ] 13. Fully charge your Robot
- [ ] 14. Keep (or put) Vector on the charging dock

Erase Vector’s data:

- [ ] 15. Place Vector on its charging station.
- [ ] 16. Press his backpack button twice.
- [ ] 17. Lift Vector’s forklift up and down.
- [ ] 18. Remove Vector from its charging station and spin one of the tank treads.
- [ ] 19. Move the tread to select **CLEAR USER DATA**.
- [ ] 20. Lift the forklift up and down to proceed.
- [ ] 21. Spin the tank treads to select **CONFIRM**. Lift the forklift up and down.

Enter Recovery Mode (aka a Factory Reset):

- [ ] 22. Place Vector on its charging station.
- [ ] 23. Hold button down until powers down completely, then for 5 seconds after the front round green light lights up. Then let the button go
- [ ] 24. Wait for Vector complete rebooting

Connect to Vector:

- [ ] 25. Open https://vector-setup.ddl.io in a Chrome browser
- [ ] 26. Place Vector in its charging station and press the backpack button twice to enter pairing mode.
- [ ] 27. Click “Pair with Vector” and select Vector from the popup.
- [ ] 28. You had to enter the PIN number into the Chrome webapp, right? If you connect, and it doesn’t ask for you to enter your PIN, you are (probably) not in recovery mode. Go back to the steps to enter recovery mode.
- [ ] 29. Verify that the emulated console is working by typing `help`

Connect to the wifi

- [ ] 30. Run, in the emulated console, `wifi-scan` to check that Vector can see your wifi
- [ ] 31. Have Vector connect to your wifi with `wifi-connect` followed by your Wifi SSID and passord

Do not close the console in chrome.

3.4 Deploy 1: Test Deploying the latest Production Software OTA image

Deploy the latest production software:

- [ ] 32. Copy the full link for `latest.ota` file from your computer (from step 12)
- [ ] 33. In the emulated console, Type `ota-start` followed by this link
- [ ] 34. Check that Vector successfully applied the production software

3.5 Deploy 2: Deploying the Unlock OTA image

When you receive your unlock OTA email, go to the next step

- [ ] 35. Download the unlock OTA image from the email into your local server folder
- [ ] 36. Check that your robot has the same ESN serial number as the one you submitted earlier, and wrote down above. The serial number is on the bottom. (https://support.digitaldreamlabs.com/article/338-how-do-i-find-vectors-serial-number) If the number does not match, you have the wrong bot.

Prepare to server the unlock OTA file .

- [ ] 37. Open https://vector-setup.ddl.io in a Chrome browser
- [ ] 38. Place Vector in its charging station and press the backpack button twice to enter pairing mode.
- [ ] 39. Click "Pair with Vector" and select Vector from the popup. Enter any Pin codes
- [ ] 40. Verify that the emulated console is working by typing `help`
- [ ] 41. Put a book or heavy block in front of Vector to keep him from driving off the charging station while updating.

Deploy the unlock OTA software:

- [ ] 42. Copy the full link for unlock `.ota` file with the QSN serial number (it has a name like `0060378a.ota` with the name)
- [ ] 43. Type `ota-start` followed by this link
- [ ] 44. Check that Vector successfully applied the unlock software

3.6 Deploy 3: Deploying the latest OSKR Software OTA image

- [ ] 45. Download the last production software from http://ota.global.anki-services.com/vic/oskr/full/latest.ota into your local server folder

Install the unlock latest OSKR file .

- [ ] 46. Open https://vector-setup.ddl.io in a Chrome browser
- [ ] 47. Place Vector in its charging station and press the backpack button twice to enter pairing mode.
- [ ] 48. Click "Pair with Vector" and select Vector from the popup. Enter any Pin codes
- [ ] 49. Verify that the emulated console is working by typing `help`

Deploy the latest OSKR software:

- [ ] 50. Copy the full link for OSKR `.ota` file
- [ ] 51. Type `ota-start` followed by this link
- [ ] 52. Check that Vector successfully applied the OSRK software
- [ ] 53. Using the mobile App, sign into Vector and add your account to him

3.7 Getting Secure Shell

- [ ] 54. Use https://vector-setup.ddl.io in a Chrome browser to connect to and pair with your Vector
- [ ] 55. Download logs from Vector
- [ ] 56. Extract the "id_rsa" key
- [ ] 57. Put the RSA key into the right place.
- [ ] 58. Get Vector's IP address
- [ ] 59. Log in using SSH

Congratulations! Vector is unlocked!

3.8 If you need to turn Vector into an Access Point to get the Key

These should only be used as necessary.

- [ ] 60. Use https://vector-setup.ddl.io in a Chrome browser to pair your with Vector
- [ ] 61. Issue the 'wifi-ap true' command
- [ ] 62. Have your computer connect to the Vector's wifi access point
- [ ] 63. Test a browser connection to http://192.168.0.2:8000

CHAPTER

FOUR

DETAILED UNLOCK STEPS

4.1 Step 1: Getting your QSN

The first step in unlocking your Vector is delivering the QSN to Digital Dream Labs so we can make your custom unlock image. This is a serial number that is included internally on the unit that is different from the ESN which you see printed on the bottom of the robot.

To get the QSN:

1. Using the Chrome web browser go to https://vector-setup.ddl.io
2. Follow the instructions to pair with Vector and log in.
3. Click the **Download logs** link located at the top right corner of the page.
4. Open the downloaded archive which should extract the contents of several logs on to your hard drive.
5. Open the file `factory/log1`. It should look like this:

```
total 13
dr--r-----    3 888      888           1024 Jan  1  1970 .
drwxr-xr-x    4 root     root          1024 Jan  1 00:00 ..
dr--r-----    2 888      888           1024 Jan  1  1970 007084d8
-r--r-----    1 888      888           1472 Jan  1  1970 AnkiRobotDeviceCert.pem
-r--r-----    1 888      888           1704 Jan  1  1970 AnkiRobotDeviceKeys.pem
-r--r-----    1 888      888            792 Jan  1  1970 Info007084d8.json
QSN=324416252 # ESN=007084d8
```

6. Deliver the QSN and ESN from the last line of the file to Digital Dream Labs.

If you have trouble extracting the archive you can send the entire .tar.bz2 file to Digital Dream Labs.

4.2 Step 2: Doing a Test Upgrade

It is **strongly recommended** that you do a test upgrade with a safe upgrade file before unlocking your robot. If the process is confusing the first time it's recommended you do a test upgrade again and again until you feel comfortable with the process.

Remember that modifying the ABOOT and recovery filesystems are dangerous operations. If the process is interrupted somehow while system files are installing it will make it impossible to boot Vector. Some potential points of failure in a normal OTA upgrade are:

- Vector loses power in the middle of the upgrade.
- Your wifi modem breaks or is unplugged in the process.

- Your service providers connection dies in the middle of the OTA file download.
- Some random part of the internet has issues preventing the OTA from downloading completely.

This doesn't matter for normal OTA installs because the process is designed to be safe and we can always fall back to the recovery software. But when unlocking we are **REWRITING** the recovery software. An interruption like this can be disastrous.

To minimize chances of network problems interfering with the update process the recommended procedure is serve your unlock image from your computer to Vector locally. This eliminates any possibility of Wide Area Network issues affecting the outcome.

The process is not particularly complex if you follow the steps as written but it can be confusing the first time you do it, particularly if you don't have much experience with networking. Luckily you can practice by uploading a normal system upgrade manually. This is *safe* and can be repeated safely.

4.2.1 Hosting the OTA locally

We need to host the OTA locally on your computer we're using to update since there will be no access to anything else at that point in time.

1. Download a local copy of the latest production firmware from http://ota.global.anki-services.com/vic/prod/full/latest.ota.
2. Create a simple webserver to serve up the files. Here we use python because it is installed on many systems and has a lightweight webserver built in. However there is nothing special about python here so you may substitute with a webserver you're more familiar with

   Assuming you saved the file in a Downloads folder:

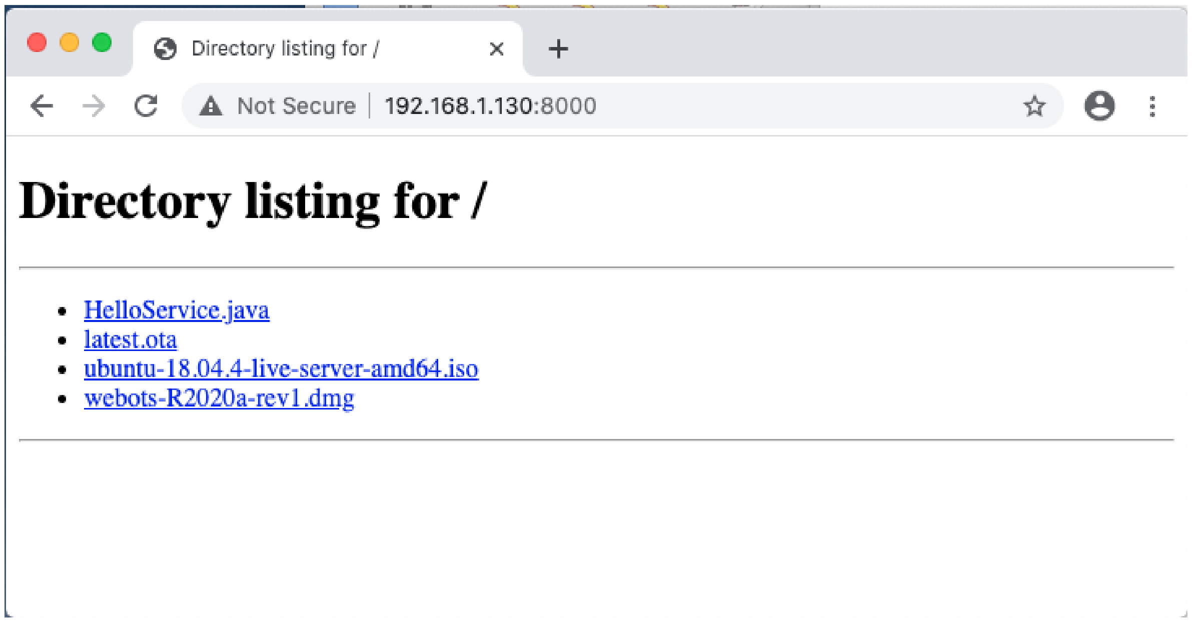
   ```
   cd ~/Downloads # or other directory
   python3 -m http.server
   ```

   Or if you don't have python3 installed:

   ```
   cd ~/Downloads # or other directory
   python -m SimpleHTTPServer
   ```

3. Determine the LAN ip of your computer. This is different from the external address and will generally start with 192.168, or 10. If you don't know how to find the address this LifeHacker article might help.
4. Open the Chrome Browser and verify that you can get to the file. If your IP address is 192.168.1.130, the link is http://192.168.1.130:8000/

   You should see a basic index page listing files including latest.ota.



Directory listing for /

- HelloService.java
- latest.ota
- ubuntu-18.04.4-live-server-amd64.iso
- webots-R2020a-rev1.dmg

5. Click on the `latest.ota` file and verify that the link works and your browser asks you to download the file.

4.2.2 Prepping the robot

First, fully charge your Vector. Keep him on the charging dock. Next wipe the user data from Vector and enter into Recovery Mode.

Erasing User Data

First wipe the user data from Vector. This will erase the entire contents of the `/data` partition. It will also give the Vector a new identity and name later.

To do so:

1. Place Vector on its charging station.
2. Press his backpack button twice. The normal BLE Pairing screen should appear on its screen.
3. Lift Vector's forklift up and down. A administration menu should appear on its screen.
4. Remove Vector from its charging station and spin one of the tank treads. This will move the > arrow pointing at an option.
5. Select **CLEAR USER DATA**. Lift the forklift up and down to proceed. A confirmation screen should appear.
6. Spin the tank treads to select **CONFIRM**. Lift the forklift up and down. Vector should reboot and start at the initial setup screen.
7. You will now need to re-attach Vector to your account via the Phone App or other means and re-download the newly generated ssh key if you wish to ssh in to Vector.

Recovery Mode (akak Factory Reset)

Next perform Recovery Mode reboot (Factory Reset) of your Vector. This will cause Vector to reboot and run using the the initial factory recovery filesystem.

To do so:

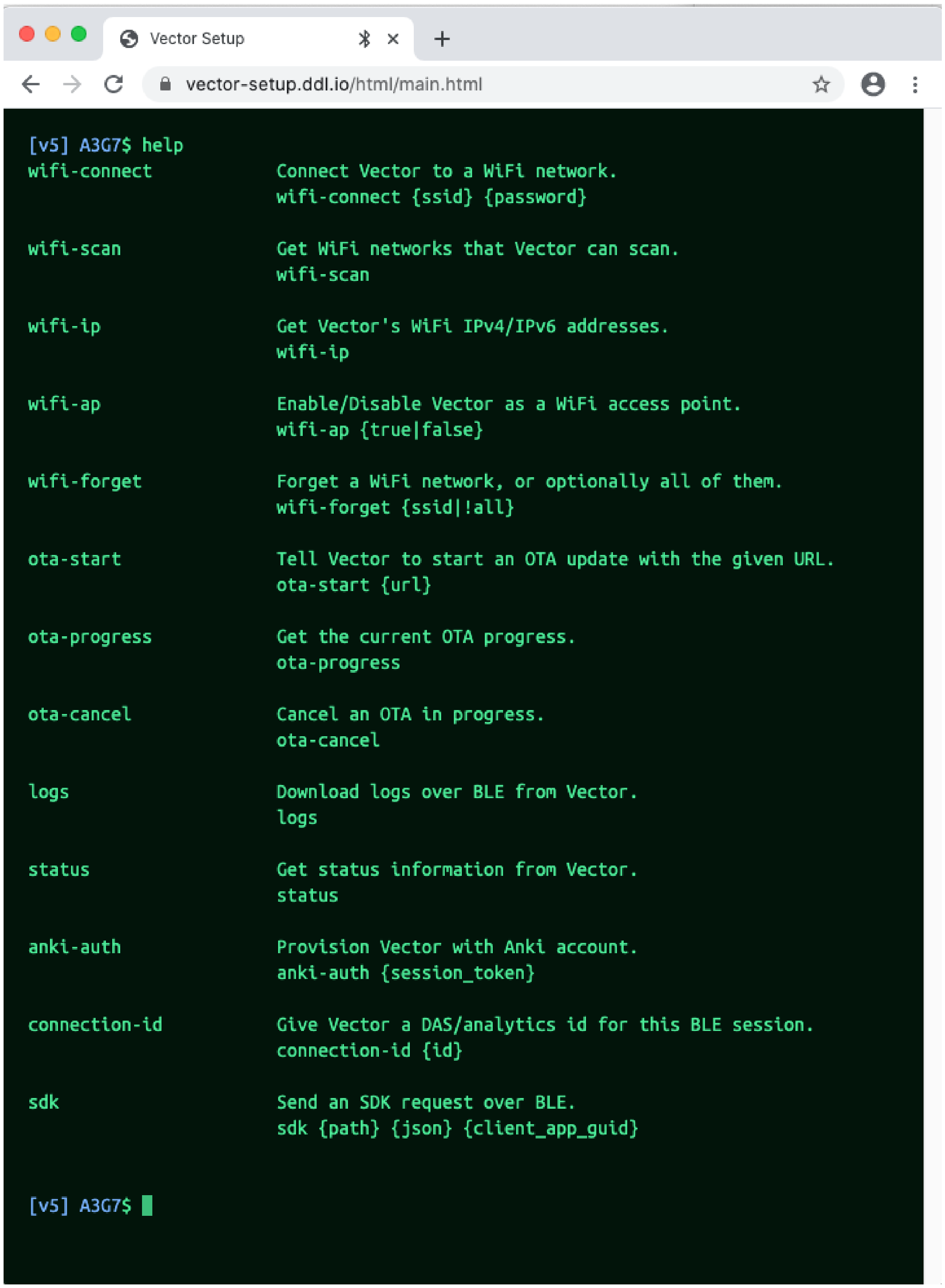
1. Place Vector on its charging station.
2. Hold its backpack button down until it powers down completely and keep holding down.
3. After approximately 5 seconds the round green light at the front of the backpack will light up. Release the button at this time.
4. Vector will reboot and start at the initial setup screen just like new.

4.2.3 Connecting to the advanced console interface in vector web setup

We will use the simulated terminal interface in Vector Web Setup to access advanced options that aren't available in the normal interface.

1. Open a new tab in Chrome and go to https://vector-setup.ddl.io
2. Place Vector in its charging station and press the backpack button twice to enter pairing mode.
3. Click "Pair with Vector" and select Vector from the popup.
4. You're now on the screen to enter the pairing code. Uncheck the box for **Enable auto-setup flow**. Enter the pin and click **Enter Pin**.
5. You will now be in an emulated terminal session in chrome. Type `help` to verify it's working.

   Keep this console open throughout the process.



```
[v5] A3G7$ help
wifi-connect            Connect Vector to a WiFi network.
                        wifi-connect {ssid} {password}

wifi-scan               Get WiFi networks that Vector can scan.
                        wifi-scan

wifi-ip                 Get Vector's WiFi IPv4/IPv6 addresses.
                        wifi-ip

wifi-ap                 Enable/Disable Vector as a WiFi access point.
                        wifi-ap {true|false}

wifi-forget             Forget a WiFi network, or optionally all of them.
                        wifi-forget {ssid|!all}

ota-start               Tell Vector to start an OTA update with the given URL.
                        ota-start {url}

ota-progress            Get the current OTA progress.
                        ota-progress

ota-cancel              Cancel an OTA in progress.
                        ota-cancel

logs                    Download logs over BLE from Vector.
                        logs

status                  Get status information from Vector.
                        status

anki-auth               Provision Vector with Anki account.
                        anki-auth {session_token}

connection-id           Give Vector a DAS/analytics id for this BLE session.
                        connection-id {id}

sdk                     Send an SDK request over BLE.
                        sdk {path} {json} {client_app_guid}


[v5] A3G7$
```

6. In the emulated terminal session, `wifi-scan` to check that Vector can see your wifi
7. Have Vector connect to your wifi with `wifi-connect` followed by your Wifi SSID and password

4.2.4 Starting the deploy

This is the critical step. We want to do this correctly.

1. Go back to your tab with the file directory listing. Since Vector will be requesting the file **DO NOT** use links with `localhost` in them or it won't be able to find the files.
2. Copy the full link for `latest.ota` to your clipboard.
3. Return to the Chrome window with the Terminal session and run `ota-start <PASTED LINK FROM ABOVE>`

If all goes well you should see a status bar update, the file should upload, Vector will reboot, and you'll have the new version of the firmware.

If it doesn't go well you may get an error status code in the window. Some of the more common status codes are:

- **203** Vector couldn't find the file. You either had the wrong link, such as using `localhost` or you don't have a webserver running.
- **216** Downgrade not allowed. You didn't do a factory reset.

4.3 Step 3: Installing the unlock image

This is the critical step. We want to do this correctly.

Now we follow the same procedure with the custom generated image for your Vector. If you skipped ahead to here *please* go through the test run with a safe file. Assuming you know what you're doing:

1. Start from an updated vector with updated software. The 0.9.0 recovery software does not support updating the appropriate partitions and it will fail to try to install the unlock image.

   Don't worry, it does this safely and you will not damage your Vector if you forget to do so.
2. Put a book or heavy block in front of Vector to keep him from driving off the charging station while updating.
3. Use the link to the custom image provided by Digital Dream Labs instead of the normal OTA file. Note that each custom image only works on ONE Vector. If you have multiple Vectors each will need its own image. Vector's face probably won't show any indication that an update is in process – he will show his eyes rather than a cloud with a spinner.

After installation is complete Vector should reboot and you should now see an introductory screen that says OSKR. This means that your Vector is working and has the new security keys needed for OSKR development.

We're out of danger now.

4.4 Step 4: Installing the latest OSKR image

Next we need to install an up-to-date software image to bring back all the expected functionality. This is a **safe** update again. If there are problems the Vector will continue to run. No need to be paranoid on this update.

Get the file http://ota.global.anki-services.com/vic/oskr/full/latest.ota and apply it to Vector the same way you applied the test upgrade. **Please Note: This is NOT the same "latest.ota" you downloaded earlier! Please download this new file and either rename it or store it elsewhere to differentiate the OSKR image from the production image you downloaded earlier.**

4.5 Sign in to Vector

You may use either the normal phone app or https://vector-setup.ddl.io to add your account to the newly imaged Vector.

4.6 Step 5: Getting your ssh key

Now that Vector is up and running and back to his normal self you'll want to get the newly created SSH key. Although this isn't strictly part of the unlock process this key is needed to access Vector's internals and begin your OSKR adventure! So lets grab it now.

To get the QSN:

1. Using the Chrome web browser go to https://vector-setup.ddl.io
2. Follow the instructions to pair with Vector and log in.
3. Click the **Download logs** link located at the top right corner of the page.
4. Open the downloaded archive which should extract the contents of several logs on to your hard drive.
5. The file `data/diagnostics/ssh/id_rsa` is your security key. It should be installed so it can be used by your system. On Linux and OSX:

   ```
   cp data/diagnostics/ssh/id_rsa_Victor-X1Y1 ~/.ssh
   chmod 700 ~/.ssh/id_rsa_Victor-X1Y1
   ```

   For Windows 10 users:

   ```
   Set-Service -Name ssh-agent -StartupType Manual
   Start-Service ssh-agent
   ssh-add id_rsa_Vector-whatever
   ssh root@192.168.whatever```
   ```

6. Load the key in to your keyring: `ssh-add ~/.ssh/id_rsa_Victor-X1Y1`
7. Obtain the IP of the robot from the diagnostics screen.
   - put Vector in charger.
   - double click the backpack button.
   - raise the forklift up and down

   You should now see the diagnostics screen which has the IP.
8. Connect! `ssh root@<ROBOT_IP_FROM_ABOVE>`
9. Type the command `exit` to leave the ssh session.

4.7 Congratulations! Vector is unlocked!

4.8 Advanced Network Configuration: Enabling AP mode

If you have a particularly bad router or wifi connection it's best to find a better connection to kick off the upgrade. However it is possible to eliminate your router as a dependency and talk directly to Vector by turning it in to an Access Point. The downside to this is that you will lose all normal internet access and only be able to talk to Vector until the process is complete.

This usually provides a better more reliable connection but some computers' network cards may have more trouble than not connecting to Vector's 2.4 Ghz network. Once again it is **strongly recommended** that you test an AP style deploy with the safe upgrade image as listed in Step 2 above.

Follow the process until bring up the simulated command terminal. From there:

1. Issue the command `wifi-ap true`. This will display new credentials for the new wifi connection.

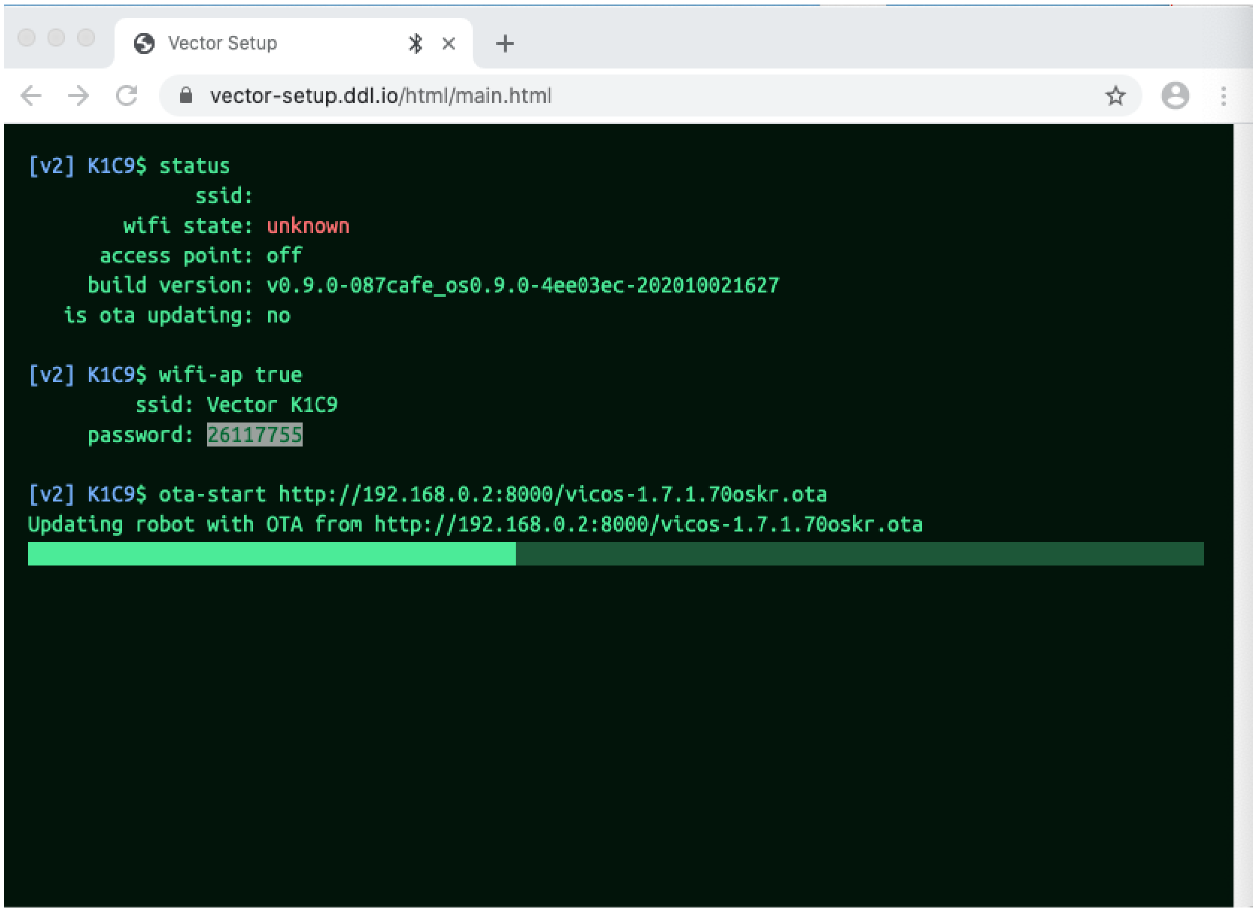
```
[v5] P7Z4$
wifi-ap true
         ssid: Vector P7Z4
   password: 20809135
```

2. Connect your computer to this access point with this password.

3. Test that you can access the file index at http://192.168.0.2:8000 which will be your new Local IP while connected to Vector.

4. Resume the existing process while still on the access point.

If all goes well Vector will install and reboot, and your computer will automatically go back to its normal network.



```
[v2] K1C9$ status
              ssid:
        wifi state: unknown
      access point: off
     build version: v0.9.0-087cafe_os0.9.0-4ee03ec-202010021627
   is ota updating: no

[v2] K1C9$ wifi-ap true
         ssid: Vector K1C9
     password: 26117755

[v2] K1C9$ ota-start http://192.168.0.2:8000/vicos-1.7.1.70oskr.ota
Updating robot with OTA from http://192.168.0.2:8000/vicos-1.7.1.70oskr.ota
```

CHAPTER

FIVE

TOOLS

5.1 Vector Self Test

Unlocking Vector adds a developer self test option to the admin menu that can be used to check all the various hardware. To access it:

1. Put Vector in its charging station.
2. Press his backpack button twice.
3. Move the forklift up and down to get to the admin menu.
4. Remove Vector from the charger, rotate his tank treads until the `>` arrow points to `RUN SELF TEST`.
5. Move the forklift up and down to select.
6. Move the tank treads to select `CONFIRM` and move the forklift up and down to begin.

Follow the instructions on screen as Vector performs the self test.

5.2 SSH / Shell Access

ssh is so integral to development that you might not think of it as a tool. But it acts as the gateway that allows access to all of Vector's internals. Without ssh you could not:

- Initiate console access to run commands.
- Modify files locally on Vector.
- Copy new files, features and enhancements, to Vector.

After obtaining the ssh key as detailed in the **Unlocking Your Robot** chapter you'll need to:

1. Load the key on your system if its not loaded. You will normally need to do this once after powering on your computer: `ssh-add ~/.ssh/id_rsa_Vector-X1Y1`
2. Obtain Vector's Internet Address from the Admin Screen. This will normally stay the same as long as you're on the same network but may change. If you suddenly can't connect re-check it.
   - Put vector in its charging station.
   - Press the backpack button twice to get to the Pairing screen.
   - Raise the forklift up and down to get to the Admin screen.
   - Get the address listed after IP.

Throughout this document we will use `<ROBOT_IP>` as a placeholder for this address. Wherever you see that replace it with this address.

3. ssh in to Vector: `ssh root@<ROBOT_IP>`

At this point you will be logged in to a Unix Shell. Once again the shell is so integral to development many developer don't think of it as a tool. But if you've never used it before there are many eccentricities you need to learn to be able to use the shell successfully. It will make your life much easier if you go through a tutorial online to get up to speed on basic usage.

5.3 Compilers

ARM compiiler

For https://anki-vic-pubfiles.anki.com/license/prod/1.0.0/licences/OStarball.v160.tgz

The https://anki-vic-pubfiles.anki.com/license/prod/1.0.0/licences/engineTarball.v160.tgz

5.4 Text editors

To edit configuration files, change settings, etc, you'll find yourself using a text editor to perform a variety of tasks. The following text editors are installed by default. If you don't know which one to use you probably want to start with **nano**.

- **nano** - The recommended editor for most users. Provides basic instructions on the screen.
- **vi** - A very low resource editor that operates in modal mode that is confusing to beginners. However it is available on the most minimal of server and embedded installs. It is useful to know the basics of vi as it will be installed on systems with no other editors.
- **mg** - A low footprint editor that mimics the basic keystrokes and opeations of the popular text editor emacs.
- **emacs** - Is not installed but aliased to the mg program for people who keep typing `emacs` out of muscle memory.

5.5 logcat

Since Vector runs a variant of the Android operating system we have access to all the standard Android tools. These are a little different than the standard linux tools. One very important one is `logcat`. This is the system used to look at all the logs being generated by various components. This includes both low level system events and high level events logged by the software controlling Vector.

To view a stream of logging in real time:

```
ssh root@<ROBOT_IP>
logcat
```

However there is so much information being logged it can be difficult to see what you want. You'll need to know two things to filter logs effectively.

First there are options to filter that are in line with a typical debugging hierarchy. You can choose a log level and only see errors that are as or more sever than the chosen log level. For example `logcat *:W` will filter out *Debug* and *Info* level messages.

- **D**ebug

- **I**information
- **W**arning
- **E**rror
- **F**atal
- **S**uppress all messages.

There is also a subsystem encoded in the logcat output. In the following example we have the subsystems *rampost*, *vic-robot*, *vic-anim*, *vic-switchboard*, *update-engine*, *vic-cloud* and *chronyd*.

```
10-09 01:58:47.525  1927  1927 I rampost : @rampost.rampost.exit000005123
10-09 01:58:48.111  1969  1969 I vic-robot: @hal.body_
↪versionfe3d1101ffffffffffffffffffffffff5773303030303030303333435613163009af999d6117621
10-09 01:58:49.751  2039  2039 I vic-anim: @random_generator.
↪seedAnimContext266417959619261
10-09 01:58:50.345  2156  2156 I vic-switchboard: @switchboard.hellohello215619854
10-09 01:58:55.172  2278  2278 I update-engine: @robot.ota_download_start000024678
10-09 01:58:55.541  2129  2169 I vic-cloud: @profile_id.start2myn3gMaZqYFjTgw9pkmnTB25051
10-09 01:59:03.799  1798  1798 I chronyd : @ntp.timesync31491
```

We can control the level of an individual subsystem and use the wildcard * to control anything that doesn't match the other parameters:

- `logcat vic-anim:* *:S` show everything for vic-anim only.
- `logcat vic-cloud:I vic-anim:I *:E` show info for vic-anim and vic-cloud and any error we see.

Just watching the logs flow by while Vector is in operation can start to give you a better idea of what the systems are doing.

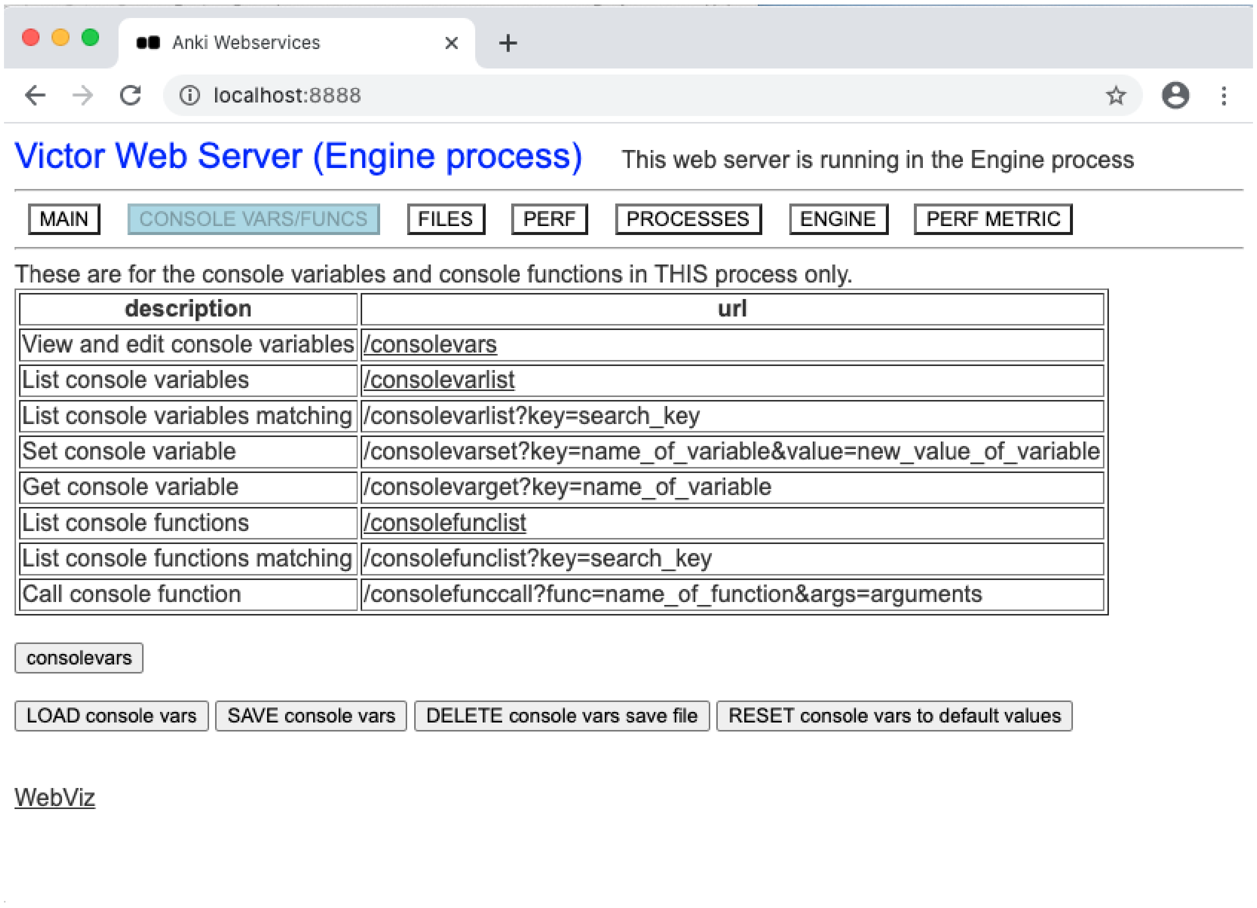
5.6 Development Web Servers

The more important Vector subsystems come with embedded webservers for development that do not run on Production Vectors. The interfaces are designed for internal use and aren't the prettiest but they provide a wealth of information about Vector.

These are enabled on OSKR Vectors, but fire-walled off by default as they allow low-level control of the Vector. There are a few ways to remove the firewall but the quickest is ssh port forwarding. Start a session with port forwarding to get access to the web servers:

```
ssh -L 8887:localhost:8887 -L 8888:localhost:8888 -L 8889:localhost:8889 root@192.168.1.
↪110
```

And now in your computer's web browser go to http://localhost:8888 to access the webserver embedded in `vic-engine`. You should see this:



Victor Web Server (Engine process) This web server is running in the Engine process

These are for the console variables and console functions in THIS process only.

description	url
View and edit console variables	/consolevars
List console variables	/consolevarlist
List console variables matching	/consolevarlist?key=search_key
Set console variable	/consolevarset?key=name_of_variable&value=new_value_of_variable
Get console variable	/consolevarget?key=name_of_variable
List console functions	/consolefunclist
List console functions matching	/consolefunclist?key=search_key
Call console function	/consolefunccall?func=name_of_function&args=arguments

consolevars
LOAD console vars
SAVE console vars
DELETE console vars save file
RESET console vars to default values
WebViz

There is **a lot** of things you can see here and you're encouraged to poke around. For now we'll provide a brief overview of the two main systems.

5.6.1 Console variables and functions

On the first screen you will see several options referring to **Console Variables**. These are used to provide various flags to change behavior of Vector, gather information, and more.

There are many many options here. Most of the options are known internally in the codebase as **Dev Cheats** and are not available on production versions of Vector software.

To get the full list of variables visit http://localhost:8888/consolevarlist. To get the full list of functions visit http://localhost:8888/consolefunclist. Examples of changing a console variable value and executing a console function can be found in the Examples section of this manual.

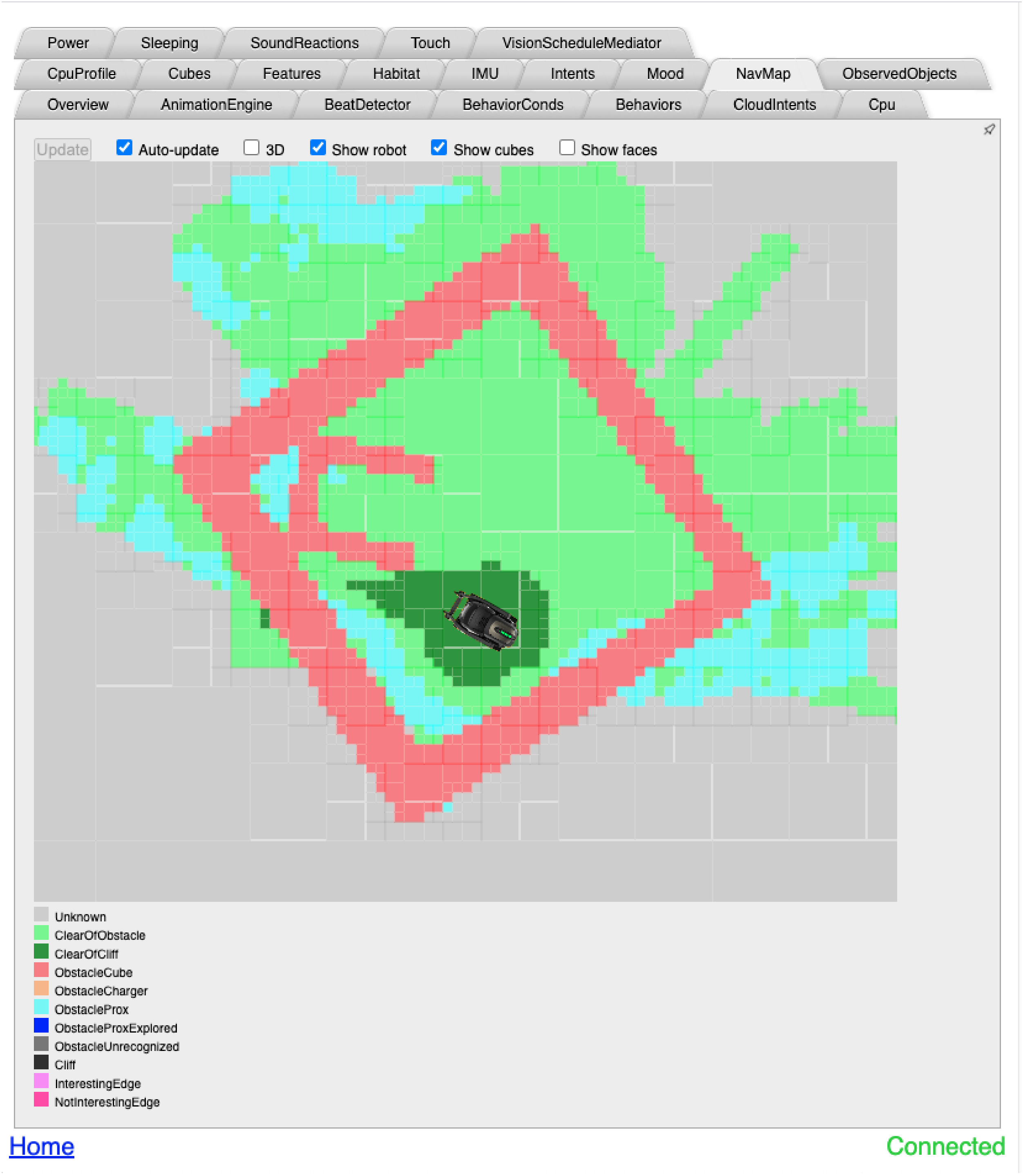
5.6.2 WebViz

WebViz provides insight in to the way Vector is operating at a high level in real time. Among other things it will show you:

- **NavMap** Vector's understanding of the enviroment around him, where table edges are located, walls, charger, obstacles.
- **CloudIntents** How Vector turns what you say into an action on its part.
- **Mood** Is Vector Happy? Confident? Social? Stimulated? Trusting?
- **Behavior** What is Vector doing now and why?

There is a link for WebViz at the bottom of the main webserver page. It can also be accessed directly at http://localhost:8888/webViz.html. Explore around and see what you can learn.

WebViz Navigation. Vector knows it's in a Vector Space



[✓] Auto-update [ ] 3D [✓] Show robot [✓] Show cubes [ ] Show faces

⬜ Unknown
🟩 ClearOfObstacle
🟩 ClearOfCliff
🟥 ObstacleCube
🟧 ObstacleCharger
🟦 ObstacleProx
🟦 ObstacleProxExplored
⬛ ObstacleUnrecognized
⬛ Cliff
🟪 InterestingEdge
🟪 NotInterestingEdge

Connected

CHAPTER

SIX

FILESYSTEM LAYOUT

A brief overview follows of the filesystem as it exists at runtime. This does not discuss the various partitions and slots used to boot the robot.

6.1 system partition

The system partition contains all the code and configuration to run the underlying variant of the Android operating system and the Userspace Anki code that powers Vector.

When you perform an upgrade of the Operating System the system partition is completely overwritten and all changes are lost.

By default it is read-only and expected to be unmodifiable.

6.2 /anki directory

This is not strictly a partition but it is where all of the Vector specific code is located. Various services, utilities, configuration settings that are not relevant to the Operating System but to Vector are located here. If you're trying to track down something that you wouldn't expect to find on a normal Linux operating system it is probably located here.

Inside you will see some familiar folders such as `/anki/etc/` `/anki/bin` and others that follow the general configuration of a Linux operating system as well as a `/data/` folder which holds various thing that don't quite fit in with the normal file layout.

6.3 /data partition

The `/data` partition is the only partition that is mounted read-write on a production Vector. All user settings are saved here. When an system upgrade is installed the data partition remains unmodified and intact and all old data is available for use by the new version of the Operating System.

This is where you will want to put any files, scripts, executables, etc that you want to survive a system upgrade.

6.4 Editing the system partition

Although the system partition is initially mounted read only it is trivial to mount it in read-write mode since you are the root user. To do so just run:

```
mount -o remount,rw /
```

And you will now be able to edit the filesystem.

CHAPTER

SEVEN

VECTOR SUBSYSTEMS

The Vector software consists of several interacting subsystems. The internal codename for the project was **Victor** and we will see that reflected in the naming with things like `vic-`.

As the software is open sourced there will be more detailed descriptions of each component.

7.1 vic-switchboard

This is the process that handles BLE communications and forwards them to the rest of the system.

7.2 vic-cloud

This is the subsystem that interacts with cloud services. Most importantly this provides access to the voice recognition functionality.

7.3 vic-anim

This controls the animations that can be played on the robot. In this case animations are a combination of motors moving and what the display shows. If Vector is dancing and smiling all the work is happening here.

7.4 vic-gateway

The internet gateway which authenticates and routes SDK messages to the vector.

7.5 vic-engine

This is the most elaborate and powerful module by far. The engine provides the high level functions for Vectors behavior, a complete AI engine that controls his mood, and much much more.

7.6 vic-dasmgr

This controls fleet logging via an ANKI developed system called DAS so we can keep an eye on the health of all the robots out there in the wild.

CHAPTER

EIGHT

SECURITY

By default the system comes in a somewhat locked down state that prevents random access by most users to the internals. This section describes several security features and settings, as well as ways to modify the settings to fit your needs.

8.1 BLE

All initial connections to Vector are made via BLE (BlueTooth Low Energy) connections via the phone app, a client tool, Vector Web Setup. These all require you to *pair* the device by entering a code displayed on the Vector screen.

Due to the limited range of BlueTooth signals and the even more limited range of being able to see the Vector's screen this does a very good job of assuring that the person accessing the Vector is in physical control of the device. Malicious users from around the world simply can't interface with the Vector at this level.

8.2 Securing SSH

By default the Vector generates a unique SSH user key the first time it boots and every time you perform a User Data Reset.

SSH works by providing a *private key* that performs encryption operations and a *public key* that allows another person or system to verify the encryption operations. As the name indicates the *private key* should only be given to someone who you want to be able sign in to the computer and the *public key* is safe to distribute to anyone without compromising the private key's security.

The default configuration is generally secure although you may wish to take some steps to harden the ssh connection even more.

8.2.1 Adding a passphrase to your local ssh key

By default the generated ssh key doesn't have a passphrase and anyone with access to the file can use it. To add a passphrase:

```
ssh-keygen -p -f ~/.ssh/id_rsa-Vector-X1Y2
```

8.2.2 Removing the private key from Vector

A user with physical access to Vector can get the private key. Once you've retrieved a copy you no longer need to keep one on the Vector. Removing the file will prevent a user from being able to download it.

```
ssh root@<ROBOT_IP>
root@Vector-U9D5:~# ls /data/ssh/
authorized_keys        id_rsa_Vector-U9D5      id_rsa_Vector-U9D5.pub
root@Vector-U9D5:~# rm /data/ssh/id_rsa_Vector-U9D5
root@Vector-U9D5:~# ls /data/ssh/
authorized_keys        id_rsa_Vector-U9D5.pub
```

If you do this you should backup your local copy of the key. If it is lost you'll need to reset User Data to generate a new key and lose any work on the `/data` partition.

8.2.3 Removing the Digital Dream Labs dev key

Digital Dream Labs includes a static ssh key to make internal development and testing easier. Although we respect users' privacy and would never connect to a customer robot there is always the possibility that the key gets out in the wild where another party can use it.

To remove the key connect to Vector and open the file `/data/ssh/authorized_keys` in a text editor. Delete the line that starts with `ssh-rsa` and ends with `digital_dream_labs_dev_key`.

8.3 Unsecuring Development Tools

There are several development tools that live behind a firewall on the robot as they allow any user with access to do a variety of things on the robot. Since the tools were originally intended for use in a development environment only they have lower levels of security than is ideal.

8.3.1 Temporary access via SSH Port Forwarding

ssh provides options to do *Port Forwarding* to create tunnels that go from your computer, over the secure connection, to the system you're logging in to. This can be used to get temporary access to a developer tool. Although not as convenient as permanently changing firewall rules, when you disconnect the ssh connection security is automatically restored.

Example accessing the Anim process webViz server:

1. ssh in to the device with Port Forwarding options:

   ```
   grant@lord-humungus otas % ssh -L8889:localhost:8889 root@192.168.1.110
   root@Vector-U9D5:~#
   ```

2. Visit http://localhost:8889 on your computer.

Multiple ports can be forwarded. Here we forward two webViz connections:

```
ssh  -L8888:localhost:8888 -L8889:localhost:8889  root@<ROBOT_IP>
```

8.3.2 Permanent access via iptables

The firewall rules are contained in the files `/etc/iptables/iptables.rules` and `/etc/iptables/ip6tables.rules` Most users won't be running IPv6 will only need to edit the first file.

The various exposed ports are already listed in the file but are commented out so they are not loaded by default:

```
# adb-over-tcp
# -A INPUT -p tcp -m tcp -m tcp --dport 5555 -j ACCEPT

# rsync (dev-deployment)
# -A INPUT -p tcp -m tcp -m tcp --dport 1873 -j ACCEPT

# webots
# -A INPUT -p tcp -m tcp --dport 5103 -j ACCEPT
# -A INPUT -p udp -m udp --dport 5103 -j ACCEPT

# dev-webservices
# -A INPUT -p tcp -m tcp -m multiport --dports 8887,8888,8889,8890 -j ACCEPT

# wwise profiler
# -A INPUT -p tcp -m tcp -m multiport --dports 24024,24025,24026 -j ACCEPT
# -A INPUT -p udp --dport 24024 -j ACCEPT
```

To open up a resource permanently:

1. Mark the system filesystem as read/write. `mount -o remount,rw /`
2. Open the file `/etc/iptables/iptables.rules` in your editor of choice.
3. delete the # sign comment from the applicable rule.
4. Restart iptables to load the rules: `systemctl restart iptables`

8.4 Manifest signing keys

By default Vector has security checks that will only allow an .ota file that is signed by Digital Dream Labs to be installed. As people in the community make their own modifications they can make their own .otas. To be able to install a community-created ota you will need to whitelist the creator's signing key. To do so:

1. On your host computer ensure the appropriate directory is created: `sudo root@<ROBOT_IP> mkdir /data/etc/ota_keys`
2. Obtain and verify the public signing key from the creator of the OTA.
3. Copy the key to vector: `scp new_key.pub root@<ROBOT_IP>:/data/etc/ota_keys`

You will now be able to install OTAs signed with this key manually.

> **WARNING** An OTA file modifies the entire operating system and has full control of Vector, and can perform very destructive actions either intentionally or unintentionally. Only install OTAs from sources you trust.

CHAPTER

NINE

EXAMPLE CUSTOMIZATION TASKS

9.1 Disable Automatic updates

If you're making changes to the system partition you'll want to disable automatic upgrades so you don't lose your work.

1. Examine logcat contents to see that we are trying to do updates every hour:

```
root@Vector-U9D5:~# logcat update-engine:* *:S
--------- beginning of main
10-07 05:13:09.568  2214  2214 I update-engine: @robot.ota_download_start000025820
10-07 05:13:10.637  2249  2249 I update-engine: @robot.ota_download_endfail1.7.1.
↪68oskrFailed to open URL: HTTP Error 403: Forbidden20300026887
```

2. Examine the contents of the update file located at `/anki/etc/update-engine.env`

```
root@Vector-U9D5:~# cat /anki/etc/update-engine.env
UPDATE_ENGINE_ENABLED=True
UPDATE_ENGINE_ALLOW_DOWNGRADE=False
UPDATE_ENGINE_BASE_URL=https://ota.global.anki-services.com/vic/prod/
UPDATE_ENGINE_ANKIDEV_BASE_URL=https://ota.global.anki-dev-services.com/vic/master/
↪lo8awreh23498sf/
```

3. Note that the `UPDATE_ENGINE_ENABLED` is set to `True`.
4. Mount the system filesystem for editing: `mount -o remount,rw /`
5. Open `/anki/etc/update-engine.env` in your text editor of choice and change the value to `False`. Save and close.
6. Reboot to ensure changes take effect: `/sbin/reboot`
7. Examine logcat contents to verify we aren't doing updates automatically:

```
root@Vector-U9D5:~# logcat update-engine:* *:S
--------- beginning of main
```

   There are no longer entries from update-engine in the logs.

9.2 Change Boot Animation

It's easy to replace the boot animation with one you like better. You will need to convert an animated gif to raw format and copy to the robot. For this you will need:

- A working installation of python with the Pillow package installed.
- An animated .gif with a resolution of 184x96 pixels
- The script `gif_to_raw.py` to convert the .gif to a raw image.

Both the script and a sample image called `bootscreen.gif` are located in the github repository for this document under the `examples/change_boot_anim` directory.

In this example we'll be operating on your HOST computer and doing everything from there. We will not be issuing commands directly on the robot.

1. Convert the .gif to a raw image:

   ```
   python gif_to_raw.py bootscreen.gif
   ```

   This will create a new file `bootscreen.gif.raw`

2. Mount the filesystem for writing. Here we'll do that from the host system:

   ```
   ssh root@192.168.1.110 "mount -o remount,rw /"
   ```

3. Use `scp` to copy the file in to place:

   ```
   scp bootscreen.gif.raw root@192.168.1.110:/anki/data/assets/cozmo_resources/config/
   engine/animations/boot_anim.raw
   ```

4. Reboot Vector from the host system:

   ```
   ssh root@192.168.1.110 "/sbin/reboot"
   ```

9.3 Edit Phrases

With this task, we can edit what Vector says in different situations- for example, we can change how he describes each weather condition, how he responds to different situations in BlackJack games, or even when he does (or doesn't) recognize a human face! Let's take a look at how to do this.

You will need:

- SSH Access to Vector

1. Once you've logged in via SSH as root, we'll need to make sure we can change files. Remount the partition as rewritable:`mount -o remount rw /`
2. Now that we can change files, it's time to see what options we have to change for the localized strings. Move into the directory where the localized strings are contained:`cd /anki/data/assets/cozmo_resources/assets/LocalizedStrings/en-US`

Once there, you'll see 2 files we can edit:

- BehaviorStrings.json
- BlackJackStrings.json

1. I recommend making backups of each file. Use these commands:`cp BehaviorStrings.json BehaviorStrings.json.bakcp BlackJackStrings.json BlackJackStrings.json.bak`(If you ever have to restore a file, use `cp` the other way around. Example: `cp BehaviorStrings.json.bak BehaviorStrings.json`)

2. Of the 2 files available here, let's open a file for editing using `nano`. Example:`nano BehaviorStrings.json`
3. Once the file is open, you will be able to see text entries that define the speech that should happen when certain weather patterns, recognition phases, or other items play. Here is an example:

```
"BehaviorDisplayWeather.Sunny": {
    "translation": "{0} degrees and sunny"
  },
```

1. Let's say we want to change this to something a bit more personal. Feel free to make your own saying:

```
"BehaviorDisplayWeather.Sunny": {
    "translation": "{0} degrees and great for a motorcycle ride"
  },
```

1. Be sure to change the other sayings for each weather type to match the same "theme" - or mix and match! You can also make edits to any other phrases you wish here, but be sure not to destroy any of the quotation marks around the phrases. Once you are finished making changes, we need to save the file with the following steps:
2. Exit the editor: `Ctrl+X`
3. Confirm the changes: `y`
4. Press `Enter` to confirm that you want to keep the same filename.
5. Reboot the robot:`sudo reboot`

Now you can test your new phrases!

9.4 Customize Eye Color(s)

You've always wanted to choose the *exact* eye color that Vector uses- now you can! With this example, we're going to edit a current color to match exactly what we want (we can't add new eye colors- yet- but for now we can edit the current ones to be exactly how we want them).

You will need:

- SSH Access to Vector

1. Once you've logged in via SSH as root, we'll need to make sure we can change files. Remount the partition as rewritable:`mount -o remount rw /`
2. Navigate to the directory where the "eye_color_config.json" is stored:`cd /anki/data/assets/cozmo_resources/config/engine`
3. Create a backup copy of the "eye_color_config.json" file just in case:`cp eye_color_config.json eye_color_config.json.bak`
4. Begin editing the "eye_color_config.json" file:`nano eye_color_config.json`
5. Find the color you want to modify. **Bear in mind you will have to continue using the name of the modified color until further source code is released.**
6. Change the "Hue" and "Saturation" values on a particular color to match what you want. You may have to experiment a little to find the perfect color! Valid values range from 0.00 to 1.00. In this example, I will take orange and turn it into red. I will change this section:

```
  "OVERFIT_ORANGE" :
   { "Hue" : 0.05, "Saturation" : 0.95 },
```

to:

```
"OVERFIT_ORANGE" :
 { "Hue" : 0.97, "Saturation" : 0.97 },
```

1. Now that we've modified the values to new settings, let's save the changes:a. Exit the editor: `Ctrl+Xb`. Confirm the changes: `yc`. Press `Enter` to confirm that you want to keep the same filename.
2. Reboot the robot:`sudo reboot`

Once Vector reboots, you're ready to test your changes! In this example, I changed the color "Orange", so I will say "Hey Vector, make your eyes orange." Vector's eyes will now actually turn to red.

Once further code is released, you'll be able to add your own colors. Escape Pod users can already change the command to fit whatever color they changed!

9.5 Watch Face Recognition via Console Variables

Here we'll use console variables to test out the face recognition of Vector.

1. Open up the firewall to provide access to the `vic-engine` webserver if you haven't already. To do this quickly `ssh -L 8888:localhost:8888 root@<ROBOT_IP>`
2. Verify the page is up by going to http://localhost:8888
3. Look at all available console variables: http://localhost:8888/consolevarlist
4. Here we're interested in `MirrorMode`
5. Refer to the main page on the webserver for instructions on how to set a console variable.

   You can type this in directly to your web browser but most professional developers would use a CLI tool like `curl` or `wget`. On your computer, and **NOT** the Vector ssh session:

   ```
   curl "http://localhost:8888/consolevarset?key=MirrorMode&value=1"
   ```

   Note that in general software developers assume zero is the same as **off** and a non-zero value is the same as **on**. Here we use 1 to turn the setting on and will later turn it off with 0.
6. The screen of Vector should now show what it is seeing. Look at Vector and you should see a box drawn around your head. The bottom of the screen should show either `UNKNOWN` or your name if you've previously identified yourself.
7. Lets assume you haven't identified yourself. Say "Hey Vector… My Name is XXXX" Vector will perform his scanning and hopefully recognize you and say hi.

   You will now see your name when he sees your face.
8. Now that we've seen that in action lets change Vector back to his normal self:

   ```
   curl "http://localhost:8888/consolevarset?key=MirrorMode&value=0"
   ```

9.6 Simulate a crash with a console function

1. Open up the firewall to provide access to the `vic-engine` webserver if you haven't already. To do this quickly `ssh -L 8888:localhost:8888 root@<ROBOT_IP>`
2. Refer to the page http://localhost:8888/consolefunclist for a list of available functions. In this case we'll use `IntentionalAbort` to simulate a software crash and verify that Vector will do a soft reboot.

   You issue the command in directly from your web browser but most professional developers would use a CLI tool like `curl` or `wget`. On your computer, and **NOT** the Vector ssh session:

   ```
   curl "http://localhost:8888/consolefunccall?func=IntentionalAbort"
   ```

CHAPTER

TEN

OPEN SOURCE REPOSITORY LIST

The Vector Robot is a versatile, fun companion for various projects, and we encourage you to use the additional resources and code that we have created to extend Vector's functionality and to customize your robot to exactly how you want it. The documentation on these items is a work in progress and we will continue to explore new examples and use cases so you can get an idea of how to use these tools and what applications they can help with. Here is a comprehensive listing of all publicly available Vector-related repositories:

10.1 Software Development Kits

Vector has 2 main software development kits, based upon Python and Go. These SDKs can be used to create new apps that connect to Vector and utilize the robot's sensors and functionality!

Go SDKPython SDK

10.2 Animations

The Animations repositories provide tools for developers and users to create new animations and/or adjust or customize Vector's eyes and other graphics!

Vector Animations (Raw)Vector Animations (Built)

10.3 Audio

The Audio repositories provide tools for developers to modify existing sounds and create new sounds for Vector!

Vector Audio (Raw)

10.4 Cloud / Hosted Service Communications

These tools and code examples can be used to develop a custom server stack or customize Chipper to program new interactions into the robot based upon voice inquiries!

Vector CloudChipperAPIAPI Clients

10.5 Escape Pod

Escape Pod Extension

10.6 Utilities

These tools are great for developers to make quick, convenient changes to robot configurations during testing!

Vector Web SetupVector ConfiguratorVector BluetoothHughOpus Go

10.7 Documentation

Much of the documentation is held in each GitHub repository! However, we do have other guides and docs on how to use the tools herein. These docs are a work in progress.

OSKR Owner's Manual

We are welcoming contributions to these repositories and would love to see what you come up with. Thanks for using OSKR!

CHAPTER

ELEVEN

APPENDIX 1: VECTOR ERROR CODES

11.1 Ota Error Codes

Code	Reason
1	Switchboard: unknown status
2	Switchboard: OTA in progress
3	Switchboard: OTA completed
4	Switchboard: rebooting
5	Switchboard: Other OTA error
10	OS: Unknown system error
200	Unexpected .tar contents
201	Unhandled manifest version or feature
202	Boot Control HAL failure
203	Could not open URL
204	URL not a TAR file
205	Decompressor error
206	Block error
207	Imgdiff error
208	I/O error
209	Signature validation error
210	Decryption error
211	Wrong base version
212	Subprocess exception
213	Wrong serial number
214	Dev / Prod mismatch
215	Socket Timeout (network stall)
216	Downgrade not allowed
217	
218	
219	Other Exception

11.2 Runtime Error Codes

These are defined in victor/robot/include/anki/cozmo/shared/factory/fault_codes.h

Code	Reason
700	SHUTDOWN_BUTTON Shutdown caused by button push
701	SHUTDOWN_GYRO_NOT_CALIBRATING
702	SHUTDOWN_BATTERY_CRITICAL_VOLT
705	SHUTDOWN_BATTERY_CRITICAL_TEMP
800	NO_ANIM_PROCESS
801	DFU_FAILED
840	NO_CAMERA_CALIB (Must run through playpen)
850	CLOUD_CERT
851	CLOUD_TOKEN_STORE
852	CLOUD_READ_ESN
870	MIC_FR
871	MIC_FL
872	MIC_BR
873	MIC_BL
890	CLIFF_FR
891	CLIFF_FL
892	CLIFF_BR
893	CLIFF_BL
894	TOF
895	TOUCH_SENSOR
898	SPINE_SELECT_TIMEOUT
899	NO_BODY
909	STOP_BOOT_ANIM_FAILED
911	AUDIO_FAILURE
913	NO_SWITCHBOARD
914	NO_ENGINE_PROCESS
915	NO_ENGINE_COMMS
916	NO_ROBOT_PROCESS
917	NO_ROBOT_COMMS
919	SYSTEMD
921	NO_GATEWAY
923	CLOUD_PROCESS_DIED
960	IMU_FAILURE
970	WIFI_HW_FAILURE
980	CAMERA_FAILURE
981	CAMERA_STOPPED
990	DISPLAY_FAILURE

CHAPTER

TWELVE

APPENDIX 2: VECTOR MAINTENANCE TASKS

This covers several tasks used to maintain all Vectors that are not specific to OSKR.

12.1 Factory Reset

Since you are modifying the contents of Vector there will be times where you break the operating system and need to reset Vector and return to the initial factory recovery filesystem.

To do so:

1. Place Vector on its charging station.
2. Hold its backpack button down until it powers down completely and keep holding down.
3. After approximately 5 seconds the round green light at the front of the backpack will light up. Release the button at this time.
4. Vector will reboot and start at the initial setup screen just like new.
5. You'll need to upload a newer version of the software via the phone App or other means.

Note that a factory reset will **NOT** erase user data. You may or may not need to do this as well depending on how much you want to clean out old settings.

12.2 Erasing User Data

There may be times where you want to erase the user configuration on a given robot. This will erase the entire contents of the /data partition including the unique ssh key and any files or code you have placed there while developing with OSKR. It will also give the Vector a new identity and name.

To do so:

1. Place Vector on its charging station.
2. Press his backpack button twice. The normal BLE Pairing screen should appear on its screen.
3. Lift Vector's forklift up and down. A administration menu should appear on its screen.
4. Remove Vector from its charging station and spin one of the tank treads. This will move the **>** arrow pointing at an option.
5. Select **CLEAR USER DATA**. Lift the forklift up and down to proceed. A confirmation screen should appear.
6. Spin the tank treads to select **CONFIRM**. Lift the forklift up and down. Vector should reboot and start at the initial setup screen.

7. You will now need to re-attach Vector to your account via the Phone App or other means and re-download the newly generated ssh key if you wish to ssh in to Vector.

12.3 CCIS Screen

The CCIS screen shows additional information about how Vector is operating. To access it:

1. Place Vector on its charging station.
2. Press his backpack button twice. The normal BLE Pairing screen should appear on its screen.
3. Lift Vector's forklift up and down. A administration menu should appear on its screen.
4. Lower Vector's head completely then raise it up completely.
5. Press the backpack button once. You should now see a new screen.
6. Continue to press the backpack button to cycle through the screens.
7. You will eventually return to the original admin screen. Left the forklift arm up and down to exit.

12.4 Backing Up JDOCS

NOTE: This operation is done at your own risk. Modifying JDOCS file values in a robot can break things, and the only way to un-break them is a Clear User Data reset.

JDOCS houses all of Vector's "memory"- his lifetime statistics and more are stored in JavaScript files on the robot in `/data/data/com.anki.victor/persistent/jdocs/`. To back these items up via ssh:

1. Make Vector's partition writable:`mount -o remount rw /`
2. Move to the 'jdocs' folder in the /data partition:`cd /data/data/com.anki.victor/persistent/jdocs`
3. Zip up the files in this directory- you can change the name of the .tar to whatever you want, this is just an example:`tar -cvf VectorsBrain.tar *`
4. Open a new Command Prompt / Terminal window **from your own machine** and transfer the file from Vector to your own computer.Syntax: `scp -i <Vector SSH Key> root@<Vector IP Address>:/data/data/com.anki.victor/persistent/jdocs/<tar name> <destination on your local computer>`Example: `scp -i C:\Users\robbie\.ssh\id_rsa_Vector-K9W7 root@192.168.50.117:/data/data/com.anki.victor/persistent/jdocs/VectorsBrain.tar C:\Users\robbie\Desktop`

CHAPTER

THIRTEEN

APPENDIX 3: LIVING WITH WINDOWS

Vector runs on a variant of the Android Operating System which in turn is based on Linux. Because of this the entire toolchain assumes that the developer is also running a Unix based system. Most Vector Developers use Macs which run Unix under the hood, and our build systems run Ubuntu 16.04. Because of this all the documentation assumes that you are on a Unix system. This can cause problems for people using Windows-based computers. Here are some tips to help out.

13.1 PuTTY - A good ssh shell for Windows

If you're only interested in connecting to Vector and working from there you can use PuTTY to easily connect to Vector and then work from there. It is widely used and there is plenty of documentation available on the web.

One complication is that PuTTY uses a different key format than the one provided by Vector when you download its ssh key. To convert the key you downloaded to the format used by PuTTY:

1. open `PuTTYgen`.
2. Click **Conversions -> Import** and select the file `id_rsa_Victor-X1Y1`
3. Go to **File -> Save private key** to save the PuTTY version of the key.

13.2 Full Linux Environment

As we release the OSKR source code you will need a full linux environment to compile the code. There are a few options here.

13.2.1 WSL - Windows Subsystem for Linux

If you simply want a console window and can work from that WSL is a tool provided by Microsoft that allows you to run Ubuntu natively. This is the best way to run things if you want to use your normal Windows environment as much as possible.

Instructions for installing are located at https://docs.microsoft.com/en-us/windows/wsl/install-win10 We currently recommend using **Ubuntu 20.04 LTS** as the Linux distribution to install.

13.2.2 VirtualBox - Virtual Linux

VirtualBox is a free program that allows you to run entire operating systems within your existing operating system. If you want a full version of Linux with its own desktop and tools you will want to install VirtualBox and install **Ubuntu 20.04 LTS** . Refer to the VirtualBox User Manual for more details.

CHAPTER FOURTEEN

APPENDIX 4: RESOURCES

- Beginner's Guide to the Bash Terminal - A good introduction to using the shell for someone with zero shell and unix experience.
- Vector-TRM A document with extensive reverse-engineering of Vector that describes how Vector works, what his parts and files are.

CHAPTER FIFTEEN

GLOSSARY

Anki : The original creators of Vector, Cozmo, and Overdrive.

DAS : Anki's internal analytics system to measure fleet performance.

OTA : Over-the-Air update. Software update over WiFi.

CLI : Command Line Interface. Text-based command system used in the event that a Graphical User Interface is not available or not desirable.