NIST SPECIAL PUBLICATION 1800-15C

Securing Small-Business and Home Internet of Things (IoT) Devices:

Mitigating Network-Based Attacks Using Manufacturer Usage Description (MUD)

Volume C:
How-To Guides

Mudumbai Ranganathan
NIST

Steve Johnson
Ashwini Kadam
Craig Pratt
Darshak Thakore
CableLabs

Eliot Lear
Cisco

William C. Barker
Dakota Consulting

Adnan Baykal
Global Cyber Alliance

Drew Cohen
Kevin Yeich
MasterPeace Solutions

Yemi Fashina
Parisa Grayeli
Joshua Harrington
Joshua Klosterman
Blaine Mulugeta
Susan Symington
The MITRE Corporation

September 2020

DRAFT

This publication is available free of charge from:
https://www.nccoe.nist.gov/projects/building-blocks/mitigating-iot-based-ddos

DISCLAIMER

National Institute of Standards and Technology Special Publication 1800-15C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-15C, 273 pages, (September 2020), CODEN: NSPUE2

FEEDBACK

You can improve this guide by contributing feedback. As you review and adopt this solution for your own organization, we ask you and your colleagues to share your experience and advice with us.

Comments on this publication may be submitted to: mitigating-iot-ddos-nccoe@nist.gov.

Public comment period: September 16, 2020 through October 16, 2020

All comments are subject to release under the Freedom of Information Act.

National Cybersecurity Center of Excellence
National Institute of Standards and Technology
100 Bureau Drive
Mailstop 2002
Gaithersburg, MD 20899
Email: nccoe@nist.gov

NATIONAL CYBERSECURITY CENTER OF EXCELLENCE

The National Cybersecurity Center of Excellence (NCCoE), a part of the National Institute of Standards and Technology (NIST), is a collaborative hub where industry organizations, government agencies, and academic institutions work together to address businesses' most pressing cybersecurity issues. This public-private partnership enables the creation of practical cybersecurity solutions for specific industries, as well as for broad, cross-sector technology challenges. Through consortia under Cooperative Research and Development Agreements (CRADAs), including technology partners—from Fortune 50 market leaders to smaller companies specializing in information technology security—the NCCoE applies standards and best practices to develop modular, easily adaptable example cybersecurity solutions using commercially available technology. The NCCoE documents these example solutions in the NIST Special Publication 1800 series, which maps capabilities to the NIST Cybersecurity Framework and details the steps needed for another entity to re-create the example solution. The NCCoE was established in 2012 by NIST in partnership with the State of Maryland and Montgomery County, Maryland.

To learn more about the NCCoE, visit https://www.nccoe.nist.gov/. To learn more about NIST, visit https://www.nist.gov.

NIST CYBERSECURITY PRACTICE GUIDES

NIST Cybersecurity Practice Guides (Special Publication 1800 series) target specific cybersecurity challenges in the public and private sectors. They are practical, user-friendly guides that facilitate the adoption of standards-based approaches to cybersecurity. They show members of the information security community how to implement example solutions that help them align more easily with relevant standards and best practices, and provide users with the materials lists, configuration files, and other information they need to implement a similar approach.

The documents in this series describe example implementations of cybersecurity practices that businesses and other organizations may voluntarily adopt. These documents do not describe regulations or mandatory practices, nor do they carry statutory authority.

ABSTRACT

The goal of the Internet Engineering Task Force's Manufacturer Usage Description (MUD) architecture is for Internet of Things (IoT) devices to behave as intended by the manufacturers of the devices. This is done by providing a standard way for manufacturers to indicate the network communications that a device requires to perform its intended function. When MUD is used, the network will automatically permit the IoT device to send and receive only the traffic it requires to perform as intended, and the network will prohibit all other communication with the device, thereby increasing the device's resilience to network-based attacks. In this project, the NCCoE has demonstrated the ability to ensure that when an IoT device connects to a home or small-business network, MUD can be used to automatically permit

the device to send and receive only the traffic it requires to perform its intended function. This NIST Cybersecurity Practice Guide explains how MUD protocols and tools can reduce the vulnerability of IoT devices to botnets and other network-based threats as well as reduce the potential for harm from exploited IoT devices. It also shows IoT device developers and manufacturers, network equipment developers and manufacturers, and service providers who employ MUD-capable components how to integrate and use MUD to satisfy IoT users' security requirements.

KEYWORDS

access control; bootstrapping; botnets; firewall rules; flow rules; Internet of Things; IoT; Manufacturer Usage Description; MUD; network segment; onboarding; router; server; threat signaling; update server; Wi-Fi Easy Connect.

DOCUMENT CONVENTIONS

The terms "shall" and "shall not" indicate requirements to be followed strictly to conform to the publication and from which no deviation is permitted.

The terms "should" and "should not" indicate that among several possibilities, one is recommended as particularly suitable without mentioning or excluding others or that a certain course of action is preferred but not necessarily required or that (in the negative form) a certain possibility or course of action is discouraged but not prohibited.

The terms "may" and "need not" indicate a course of action permissible within the limits of the publication.

The terms "can" and "cannot" indicate a possibility and capability, whether material, physical, or causal.

Acronyms used in figures can be found in the Acronyms appendix.

CALL FOR PATENT CLAIMS

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory [ITL] draft publication). Such guidance and/or requirements may be directly stated in this ITL publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

1. assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

2. assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   a. under reasonable terms and conditions that are demonstrably free of any unfair discrimination or

   b. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to mitigating-iot-ddos-nccoe@nist.gov

ACKNOWLEDGMENTS

We are grateful to the following individuals for their generous contributions of expertise and time.

Name	Organization
Allaukik Abhishek	Arm
Michael Bartling	Arm
Mark Walker	CableLabs
Tao Wan	CableLabs
Russ Gyurek	Cisco
Peter Romness	Cisco
Brian Weis	Cisco

Name	Organization
Rob Cantu	CTIA
Dean Coclin	DigiCert
Avesta Hojjati	DigiCert
Clint Wilson	DigiCert
Katherine Gronberg	Forescout
Tim Jones	Forescout
Rae'-Mar Horne	MasterPeace Solutions, Ltd.
Nate Lesser	MasterPeace Solutions, Ltd.
Tom Martz	MasterPeace Solutions, Ltd.
Daniel Weller	MasterPeace Solutions, Ltd.
Nancy Correll	The MITRE Corporation
Sallie Edwards	The MITRE Corporation
Drew Keller	The MITRE Corporation
Sarah Kinling	The MITRE Corporation
Karri Meldorf	The MITRE Corporation
Mary Raguso	The MITRE Corporation

Name	Organization
Allen Tan	The MITRE Corporation
Mo Alhroub	Molex
Jaideep Singh	Molex
Bill Haag	NIST
Tim Polk	NIST
Murugiah Souppaya	NIST
Paul Watrobski	NIST
Bryan Dubois	Patton Electronics
Stephen Ochs	Patton Electronics
Karen Scarfone	Scarfone Cybersecurity
Matt Boucher	Symantec
Petros Efstathopoulos	Symantec
Bruce McCorkendale	Symantec
Susanta Nanda	Symantec
Yun Shen	Symantec
Pierre-Antoine Vervier	Symantec

Name	Organization
John Bambenek	ThreatSTOP
Russ Housley	Vigil Security

The Technology Partners/Collaborators who participated in this build submitted their capabilities in response to a notice in the Federal Register. Respondents with relevant capabilities or product components were invited to sign a Cooperative Research and Development Agreement (CRADA) with NIST, allowing them to participate in a consortium to build this example solution. We worked with:

Technology Partner/Collaborator	Build Involvement
Arm	Subject matter expertise
CableLabs	Micronets Gateway Micronets cloud infrastructure Prototype IoT devices–Raspberry Pi with Wi-Fi Easy Connect support Micronets mobile application
Cisco	Cisco Catalyst 3850S MUD manager
CTIA	Subject matter expertise
DigiCert	Private Transport Layer Security certificate Premium Certificate
Forescout	Forescout appliance–VCT-R Enterprise manager–VCEM-05
Global Cyber Alliance	Quad9 DNS service, Quad9 Threat Application Programming Interface ThreatSTOP threat MUD file server
MasterPeace Solutions	Yikes! router Yikes! cloud Yikes! mobile application

Technology Partner/Collaborator	Build Involvement
Molex	Molex light-emitting diode light bar Molex Power over Ethernet Gateway
Patton Electronics	Subject matter expertise
Symantec	Subject matter expertise

Contents

List of Figures

List of Tables

1 Introduction

This following volumes of this guide show information technology (IT) professionals and security engineers how we implemented this example solution. We cover all of the products employed in this reference design. We do not re-create the product manufacturers' documentation, which is presumed to be widely available. Rather, these volumes show how we incorporated the products together in our environment.

Note: These are not comprehensive tutorials. There are many possible service and security configurations for these products that are out of scope for this reference design.

1.1 How to Use this Guide

This National Institute of Standards and Technology (NIST) Cybersecurity Practice Guide demonstrates a standards-based reference design for mitigating network-based attacks by securing home and small-business Internet of Things (IoT) devices. The reference design is modular, and it can be deployed in whole or in part. This practice guide provides users with the information they need to replicate four example MUD-based implementations of this reference design. These example implementations are referred to as Builds, and this volume describes in detail how to reproduce each one.

This guide contains three volumes and a supplement:

- NIST SP 1800-15A: *Executive Summary – why we wrote this guide, the challenge we address, why it could be important to your organization, and our approach to solving this challenge*
- NIST SP 1800-15B: *Approach, Architecture, and Security Characteristics – what we built and why, including the risk analysis performed, and the security control map*
- NIST SP 1800-15C: *How-To Guides – instructions for building the example implementations including all the security relevant details that would allow you to replicate all or parts of this project* ***(you are here)***
- Functional Demonstration Results - supplement to NIST SP 1800-15B: *describes the functional demonstration results for the four implementations of the MUD-based reference solution*

Depending on your role in your organization, you might use this guide in different ways:

Business decision makers, including chief security and technology officers, will be interested in the *Executive Summary,* NIST SP 1800-15A, which describes the following topics:

- challenges that enterprises face in trying to mitigate network-based attacks by securing home and small-business IoT devices
- example solutions built at the National Cybersecurity Center of Excellence (NCCoE)
- benefits of adopting the example solutions

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in NIST SP 1800-15B, which describes what we did and why. The following sections will be of particular interest:

- Section 3.4, Risk Assessment, describes the risk analysis we performed.
- Section 5.2, Security Control Map, maps the security characteristics of these example solutions to cybersecurity standards and best practices.

You might share the *Executive Summary,* NIST SP 1800-15A, with your leadership team members to help them understand the importance of adopting a standards-based solution for mitigating network-based attacks by securing home and small-business IoT devices.

IT professionals who want to implement an approach like this will find this whole practice guide useful. You can use this How-To portion of the guide, NIST SP 1800-15C, to replicate all or parts of one or all four builds created in our lab. This How-To portion of the guide provides specific product installation, configuration, and integration instructions for implementing the example solutions. We do not re-create the product manufacturers' documentation, which is generally widely available. Rather, we show how we incorporated the products together in our environment to create an example solution.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of products to address this challenge, this guide does not endorse these particular products. Your organization can adopt one of these solutions or one that adheres to these guidelines in whole, or you can use this guide as a starting point for tailoring and implementing parts of a Manufacturer Usage Description (MUD)-based solution. Your organization's security experts should identify the products that will best integrate with your existing tools and IT system infrastructure. We hope that you will seek products that are congruent with applicable standards and best practices. NIST SP 1800-15B lists the products that we used in each build and maps them to the cybersecurity controls provided by this reference solution.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but a possible solution. In the case of this guide, it describes four possible solutions. This is a draft guide. We seek feedback on its contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to mitigating-iot-ddos-nccoe@nist.gov.

1.2 Build Overview

This NIST Cybersecurity Practice Guide addresses the challenge of using standards-based protocols and available technologies to mitigate network-based attacks by securing home and small-business IoT devices. It identifies three key forms of protection:

- use of the MUD specification to automatically permit an IoT device to send and receive only the traffic it requires to perform as intended, thereby reducing the potential for the device to be the victim of a network-based attack, as well as the potential for the device, if compromised, to be used in a network-based attack

- use of network-wide access controls based on threat intelligence to protect all devices (both MUD-capable and non-MUD-capable) from connecting to domains that are known current threats
- automated secure software updates to all devices to ensure that operating system (OS) patches are installed promptly

Four builds that serve as example solutions of how to support the MUD specification have been implemented and demonstrated as part of this project. This practice guide provides instructions for reproducing these four builds.

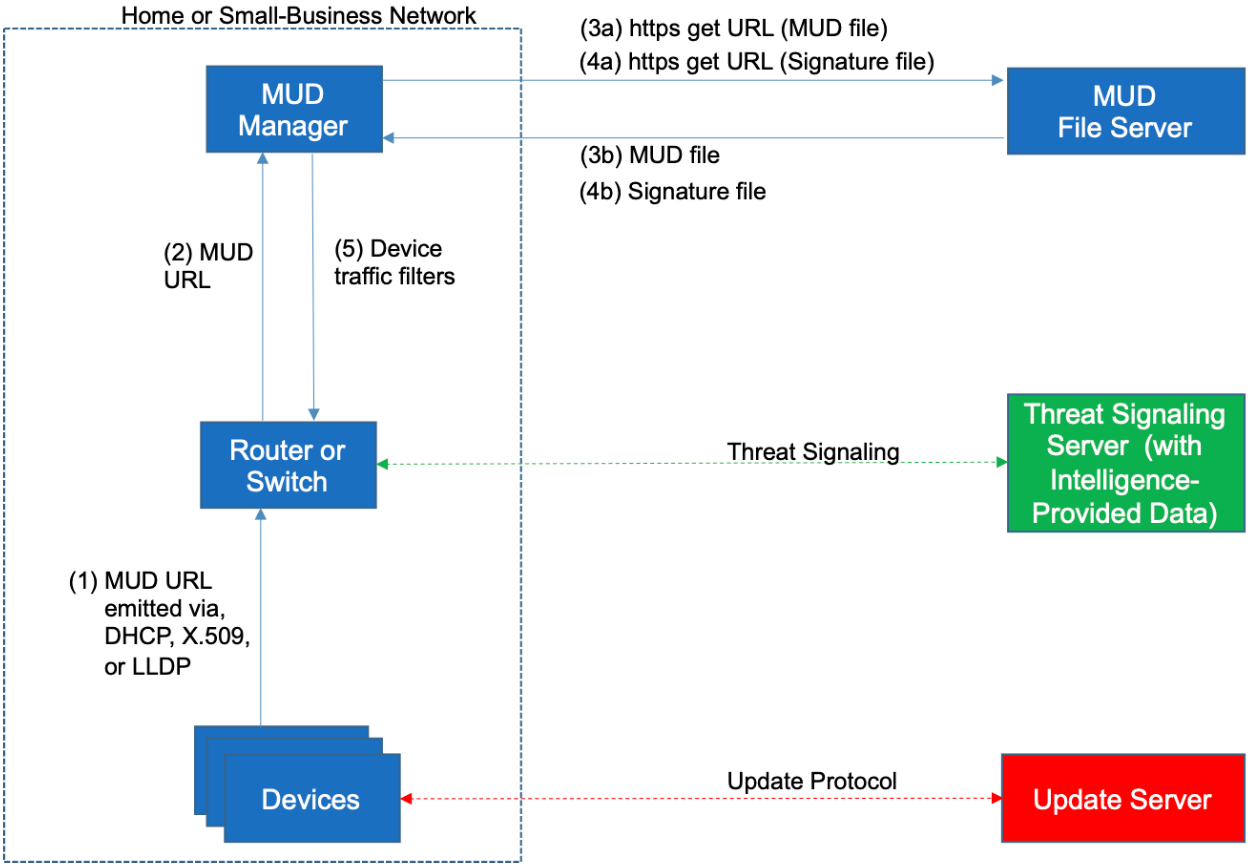
1.2.1 Usage Scenarios

Each of the four builds is designed to fulfill the use case of a MUD-capable IoT device being onboarded and used on home and small-business networks, where plug-and-play deployment is required. All four builds include both MUD-capable and non-MUD-capable IoT devices. MUD-capable IoT devices include the Molex Power over Ethernet (PoE) Gateway and Light Engine as well as four development kits (devkits) that the National Cybersecurity Center of Excellence (NCCoE) configured to perform actions such as power a light-emitting diode (LED) bulb on and off, start network connections, and power a connected lighting device on and off. These MUD-capable IoT devices interact with external systems to access notional, secure updates and various cloud services, in addition to interacting with traditional personal computing devices, as permitted by their MUD files. Non-MUD-capable IoT devices deployed in the builds include three cameras, two mobile phones, two connected lighting devices, a connected assistant, a connected printer, a baby monitor with remote control and video and audio capabilities, a connected wireless access point, and a connected digital video recorder. The cameras, connected lighting devices, baby monitor, and connected digital video recorder are all controlled and managed by a mobile phone. In combination, these devices are capable of generating a wide range of network traffic that could reasonably be expected on a home or small-business network.

1.2.2 Reference Architecture Overview

Figure 1-1 depicts a general reference design for all four builds. It consists of three main components: support for MUD, support for threat signaling, and support for periodic updates.

Figure 1-1 Reference Architecture


Home or Small-Business Network
MUD Manager
(3a) https get URL (MUD file)
(4a) https get URL (Signature file)
MUD File Server
(3b) MUD file
(4b) Signature file
(2) MUD URL
(5) Device traffic filters
Router or Switch
Threat Signaling
Threat Signaling Server (with Intelligence-Provided Data)
(1) MUD URL emitted via, DHCP, X.509, or LLDP
Devices
Update Protocol
Update Server

1.2.2.1 Support for MUD

A new functional component, the MUD manager, is introduced to augment the existing networking functionality offered by the home/small-business network router or switch. Note that the MUD manager is a logical component. Physically, the functionality it provides can and often will be combined with that of the network router or switch in a single device.

IoT devices must somehow be associated with a MUD file. The MUD specification describes three possible mechanisms through which the IoT device can provide the MUD file URL to the network: inserting the MUD URL into the Dynamic Host Configuration Protocol (DHCP) address requests that they generate when they attach to the network (e.g., when powered on), providing the MUD URL in a Link Layer Discovery Protocol (LLDP) frame, or providing the MUD URL as a field in an X.509 certificate that the device provides to the network via a protocol such as Tunnel Extensible Authentication Protocol. In addition, the MUD specification provides flexibility to enable other mechanisms by which MUD file URLs can be associated with IoT devices. One such alternative mechanism is to associate the device with its MUD file by using the device's bootstrapping information that is conveyed as part of the Wi-Fi Easy Connect (also referred to as Device Provisioning Protocol—DPP) onboarding process. This is the mechanism implemented in Build 3.

Figure 1-1 uses labeled arrows to depict the steps involved in supporting MUD:

- The IoT device emits a MUD URL by using a mechanism such as DHCP, LLDP, or X.509 certificate (step 1).
- The router extracts the MUD URL from the protocol frame of whatever mechanism was used to convey it and forwards this MUD URL to the MUD manager (step 2).
- Once the MUD URL is received, the MUD manager uses https to request the MUD file from the MUD file server by using the MUD URL provided in the previous step (step 3a); if successful, the MUD file server at the specified location will serve the MUD file (step 3b).
- Next, the MUD manager uses https to request the signature file associated with the MUD file (step 4a) and upon receipt (step 4b) verifies the MUD file by using its signature file.
- The MUD file describes the communications requirements for the IoT device. Once the MUD manager has determined the MUD file to be valid, the MUD manager converts the access control rules in the MUD file into access control entries (e.g., access control lists—ACLs, firewall rules, or flow rules) and installs them on the router or switch (step 5).

Once the device's access control rules are applied to the router or switch, the MUD-capable IoT device will be able to communicate with approved local hosts and internet hosts as defined in the MUD file, and any unapproved communication attempts will be blocked.

1.2.2.2 Support for Updates

To provide additional security, the reference architecture also supports periodic updates. All builds include a server that is meant to represent an update server to which MUD will permit devices to connect. Each IoT device on an operational network should be configured to periodically contact its update server to download and apply security patches, ensuring that it is running the most up-to-date and secure code available. To ensure that such updates are possible, the IoT device's MUD file must explicitly permit the IoT device to receive traffic from the update server. Although regular manufacturer updates are crucial to IoT security, the builds described in this practice guide demonstrate only the ability to receive faux updates from a notional update server.

1.2.2.3 Support for Threat Signaling

To provide additional protection for both MUD-capable and non-MUD-capable devices, the reference architecture also incorporates support for threat signaling. The router or switch can receive threat feeds from a threat signaling server to use as a basis for restricting certain types of network traffic. For example, both MUD-capable and non-MUD-capable devices can be prevented from connecting to internet domains that have been identified as potentially malicious.

1.2.2.4 Build-Specific Features

The reference architecture depicted in Figure 1-1 is intentionally general. Each build instantiates this reference architecture in a unique way, depending on the equipment used and the capabilities supported. The logical and physical architectures of each build are depicted and described in NIST SP 1800-15B: *Approach, Architecture, and Security Characteristics*. While all four builds support MUD and the ability to receive faux updates from a notional update server, only Build 2 currently supports threat signaling. Only Build 3 currently supports onboarding MUD-capable devices using the Wi-Fi Alliance Wi-Fi Easy Connect protocol. Build 1 and Build 2 include nonstandard device discovery technology to discover, inventory, profile, and classify attached devices. Such classification can be used to validate that the access being granted to each device is consistent with that device's manufacturer and model. In Build 2, a device's manufacturer and model can be used as a basis for identifying and enforcing that device's traffic profile.

Briefly, the four builds of the reference architecture that have been completed and demonstrated are as follows:

- Build 1 uses products from Cisco Systems, DigiCert, Forescout, and Molex. The Cisco MUD manager supports MUD, and the Forescout virtual appliances and enterprise manager perform non-MUD-related device discovery on the network. Molex PoE Gateway and Light Engine is used as a MUD-capable IoT device. Certificates from DigiCert are also used.
- Build 2 uses products from MasterPeace Solutions Ltd., Global Cyber Alliance (GCA), ThreatSTOP, and DigiCert. The MasterPeace Solutions Yikes! router, cloud service, and mobile application support MUD as well as perform device discovery on the network and apply additional traffic rules to both MUD-capable and non-MUD-capable devices based on device manufacturer and model. The GCA threat agent, Quad9 DNS service, and ThreatSTOP threat MUD file server support threat signaling. Certificates from DigiCert are also used.
- Build 3 uses products from CableLabs and DigiCert. CableLabs Micronets (e.g., Micronets Gateway, Micronets Manager, Micronets mobile phone application, and related service provider cloud-based infrastructure) supports MUD and implements the Wi-Fi Alliance's Wi-Fi Easy Connect protocol to securely onboard devices to the network. It also uses software-defined networking to create separate trust zones (e.g., network segments) called *micronets* to which devices are assigned according to their intended network function. Certificates from DigiCert are also used.
- Build 4 uses software developed at the NIST Advanced Networking Technologies laboratory. This software supports MUD and is intended to serve as a working prototype of the MUD request for comments (RFC) to demonstrate feasibility and scalability. Certificates from DigiCert are also used.

The logical architectures and detailed descriptions of Builds 1, 2, 3, and 4 can be found in NIST SP 1800-15B: *Approach, Architecture, and Security Characteristics*.

1.2.3 Physical Architecture Overview

Figure 1-2 depicts the high-level physical architecture of the NCCoE laboratory environment. This implementation currently supports four builds and has the flexibility to implement additional builds in the future. As depicted, the NCCoE laboratory network is connected to the internet via the NIST data center. Access to and from the NCCoE network is protected by a firewall. The NCCoE network includes a shared virtual environment that houses an update server, a MUD file server, an unapproved server (i.e., a server that is not listed as a permissible communications source or destination in any MUD file), a Message Queuing Telemetry Transport (MQTT) broker server, and a Forescout enterprise manager. These components are hosted at the NCCoE and are used across builds where applicable. The Transport Layer Security (TLS) certificate and Premium Certificate used by the MUD file server are provided by DigiCert.

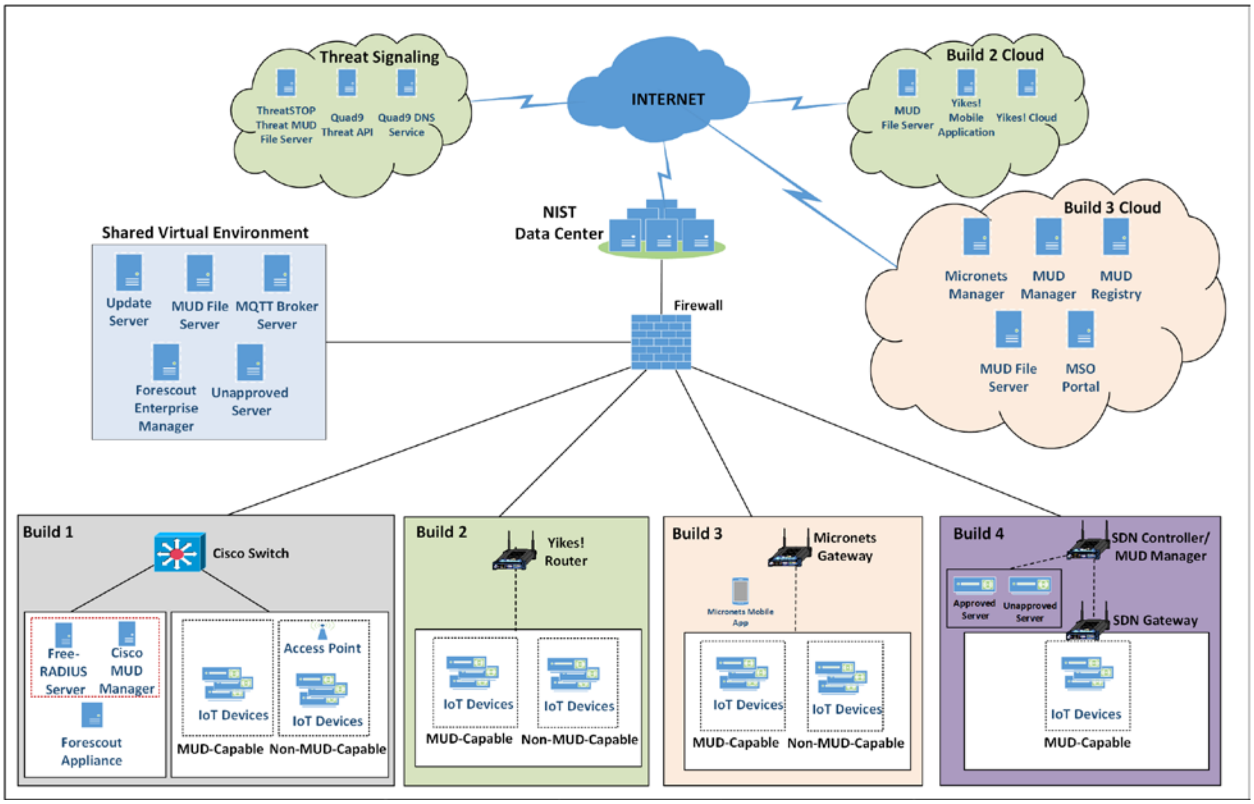
The following four builds, as depicted in the diagram, are supported within the physical architecture:

- Build 1 network components consist of a Cisco Catalyst 3850-S switch, a Cisco MUD manager, a FreeRADIUS server, and a virtualized Forescout appliance on the local network. Build 1 also requires support from all components that are in the shared virtual environment, including the Forescout enterprise manager.
- Build 2 network components consist of a MasterPeace Solutions Ltd. Yikes! router on the local network. Build 2 requires support from the MUD file server, Yikes! cloud, and a Yikes! mobile application that are resident on the Build 2 cloud. The Yikes! router includes threat-signaling capabilities (not depicted) that have been integrated with it. Build 2 also requires support from threat-signaling cloud services that consist of the ThreatSTOP threat MUD file server, Quad9 threat application programming interface (API), and Quad9 DNS service. Build 2 uses only the update server and unapproved server components that are in the shared virtual environment.
- Build 3 network components consist of a CableLabs Micronets Gateway/wireless access point (AP). The Gateway/wireless AP resides on the local network and operates in conjunction with various service provider components and partner/service provider offerings that reside in the Micronets virtual environment in the Build 3 cloud. The Micronets Gateway is controlled by a Micronets Manager that resides in the Build 3 cloud and that coordinates a number of cloud-based Micronets micro-services, some of which are depicted. Build 3 also includes a Micronets mobile application that provides the user and device interfaces for device onboarding.
- Build 4 network components consist of a software-defined networking (SDN)-capable gateway/switch on the local network and an SDN controller/MUD manager and approved and unapproved servers that are located remotely from the local network. Build 4 also uses the MUD file server that is resident in the shared virtual environment.

IoT devices used in all four builds include both MUD-capable and non-MUD-capable IoT devices. The MUD-capable IoT devices used, which vary across builds, include Raspberry Pi, ARTIK, u-blox, Intel UP Squared, BeagleBone Black, NXP i.MX 8M (devkit), and the Molex Light Engine controlled by PoE Gateway. Non-MUD-capable devices used, which also vary across builds, include a wireless access point, cameras, a printer, mobile phones, lighting devices, a connected assistant device, a baby monitor, and a digital video recorder. Each of the completed builds and the roles that their components play in their architectures are explained in more detail in NIST SP 1800-15B.

The remainder of this guide describes how to implement Builds 1, 2, 3, and 4.

Figure 1-2 NCCoE Physical Architecture



1.3 Typographic Conventions

The following table presents typographic conventions used in this volume.

Typeface/Symbol	Meaning	Example
Italics	file names and path names; references to documents that are not hyperlinks; new terms; and placeholders	For language use and style guidance, see the *NCCoE Style Guide*.
Bold	names of menus, options, command buttons, and fields	Choose **File > Edit.**
`Monospace`	command-line input, onscreen computer output, sample code examples, and status codes	`Mkdir`
`Monospace Bold`	command-line user input contrasted with computer output	**`service sshd start`**
blue text	link to other parts of the document, a web URL, or an email address	All publications from NIST's NCCoE are available at https://www.nccoe.nist.gov.

2 Build 1 Product Installation Guides

This section of the practice guide contains detailed instructions for installing and configuring all the products used to implement Build 1. For additional details on Build 1's logical and physical architectures, please refer to NIST SP 1800-15B.

2.1 Cisco MUD Manager

This section describes how to deploy Cisco's MUD manager version 1.0, which uses a MUD-based authorization system in the network, using Cisco Catalyst switches, FreeRADIUS, and Cisco MUD manager.

2.1.1 Cisco MUD Manager Overview

The Cisco MUD manager is an open-source implementation that works with IoT devices that emit their MUD URLs. In this implementation we tested two MUD URL emission methods: DHCP and LLDP. The MUD manager is supported by a FreeRADIUS server that receives MUD URLs from the switch. The MUD URLs are extracted by the DHCP server and are sent to the MUD manager via Remote Authentication Dial-In User Service (RADIUS) messages. The MUD manager is responsible for retrieving the MUD file and corresponding signature file associated with the MUD URL. The MUD manager verifies the legitimacy of the file and then translates the contents to an internet protocol (IP) ACL-based policy that is installed on the switch.

The version of the Cisco MUD manager used in this project is a proof-of-concept implementation that is intended to introduce advanced users and engineers to the MUD concept. It is not a fully automated MUD manager implementation, and some protocol features are not present. At implementation, the “model” construct was not yet implemented. In addition, if a DNS-based system changes its address, this will not be noticed. Also, IPv6 access has not been fully supported.

2.1.2 Cisco MUD Manager Configurations

The following subsections document the software, hardware, and network configurations for the Cisco MUD manager.

2.1.2.1 Hardware Configuration

Cisco requires installing the MUD manager and FreeRADIUS on a single server with at least 2 gigabytes of random access memory. This server must integrate with at least one switch or router on the network. For this build we used a Catalyst 3850-S switch.

2.1.2.2 Network Configuration

The MUD manager and FreeRADIUS server instances were installed and configured on a dedicated machine leveraged for hosting virtual machines in the Build 1 lab environment. This machine was then connected to virtual local area network (VLAN) 2 on the Catalyst 3850-S and assigned a static IP address.

2.1.2.3 Software Configuration

For this build, the Cisco MUD manager was installed on an Ubuntu 18.04.01 64-bit server. However, there are many approaches for implementation. Alternatively, the MUD manager can be built via docker containers provided by Cisco.

The Cisco MUD manager can operate on Linux operating systems, such as

- Ubuntu 18.04.01
- Amazon Linux

The Cisco MUD manager requires the following installations and components:

- OpenSSL
- cJSON
- MongoDB
- Mongo C driver

- Libcurl
- FreeRADIUS server

At a high level, the following software configurations and integrations are required:

- The Cisco MUD manager requires integration with a switch (such as a Catalyst 3850-S) that connects to an authentication, authorization, and accounting (AAA) server that communicates by using the RADIUS protocol (i.e., a RADIUS server).
- The RADIUS server must be configured to identify a MUD URL received in an accounting request message from a device it has authenticated.
- The MUD manager must be configured to process a MUD URL received from a RADIUS server and return access control policy to the RADIUS server, which is then forwarded to the switch.

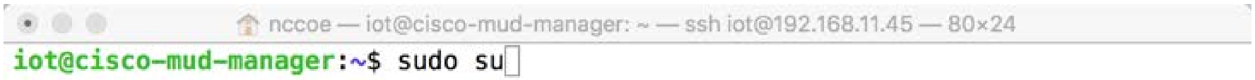
2.1.3 Setup

2.1.3.1 Preinstallation

Cisco's DevNet GitHub page provides documentation that we followed to complete this section: https://github.com/CiscoDevNet/MUD-Manager/tree/3.0.1#dependancies

1. Open a terminal window, and enter the following command to log in as root:

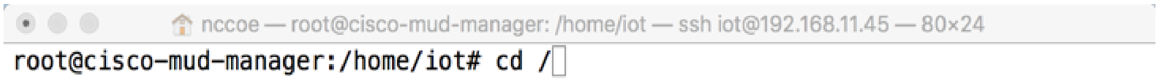
   ```
   sudo su
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo su
```

2. Change to the root directory:

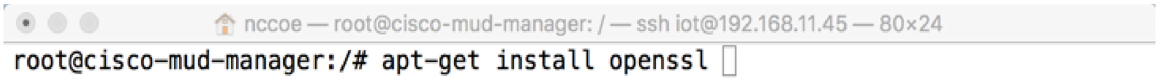
   ```
   cd /
   ```


nccoe — root@cisco-mud-manager: /home/iot — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/home/iot# cd /
```

3. To install OpenSSL from the terminal, enter the following command:

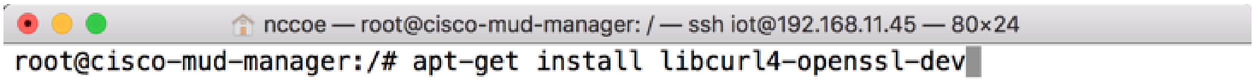
   ```
   apt-get install openssl
   ```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# apt-get install openssl
```

   a. If unable to link to OpenSSL, install the following by entering this command:

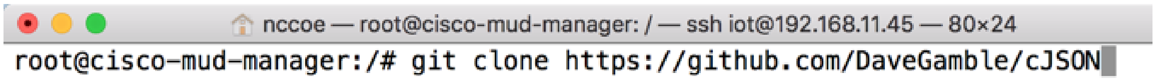
   ```
   apt-get install -y libssl-dev
   ```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# apt-get install libcurl4-openssl-dev
```

4. To install cJSON, download it from GitHub by entering the following command:

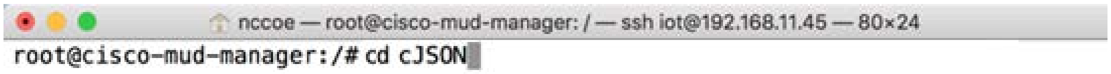
   ```
   git clone https://github.com/DaveGamble/cJSON
   ```

   
   nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

   ```
   root@cisco-mud-manager:/# git clone https://github.com/DaveGamble/cJSON
   ```

   a. Change directories to the cJSON folder by entering the following command:

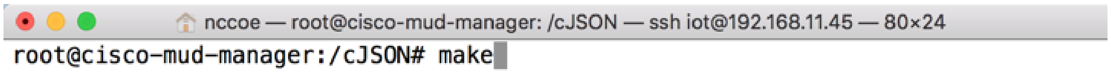
      ```
      cd cJSON
      ```

      
      nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

      ```
      root@cisco-mud-manager:/# cd cJSON
      ```

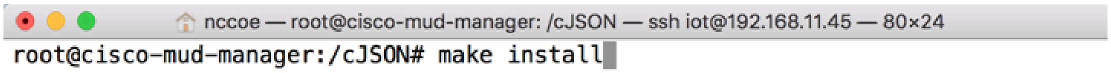
   b. Build cJSON by entering the following commands:

      ```
      make
      ```

      
      nccoe — root@cisco-mud-manager: /cJSON — ssh iot@192.168.11.45 — 80×24

      ```
      root@cisco-mud-manager:/cJSON# make
      ```

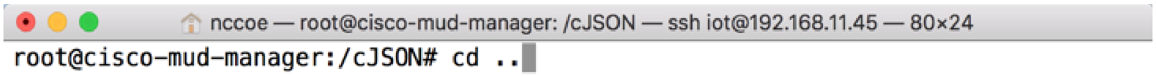
      ```
      make install
      ```

      
      nccoe — root@cisco-mud-manager: /cJSON — ssh iot@192.168.11.45 — 80×24

      ```
      root@cisco-mud-manager:/cJSON# make install
      ```

5. Change directories back a folder by entering the following command:

   ```
   cd ..
   ```

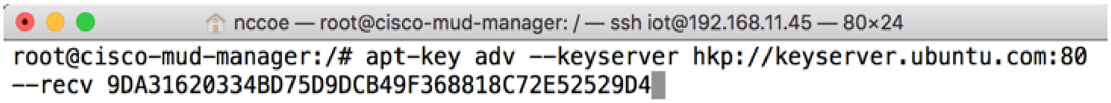
   
   nccoe — root@cisco-mud-manager: /cJSON — ssh iot@192.168.11.45 — 80×24

   ```
   root@cisco-mud-manager:/cJSON# cd ..
   ```

6. To install MongoDB, enter the following commands:

   a. Import the public key:

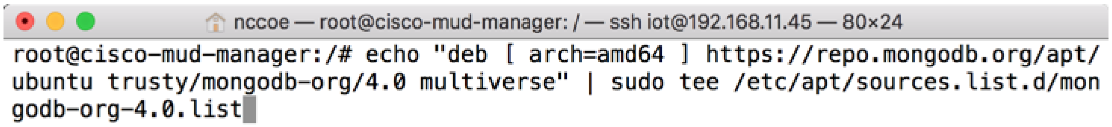
      ```
      apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
      9DA31620334BD75D9DCB49F368818C72E52529D4
      ```

      
      nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

      ```
      root@cisco-mud-manager:/# apt-key adv --keyserver hkp://keyserver.ubuntu.com:80
      --recv 9DA31620334BD75D9DCB49F368818C72E52529D4
      ```

   b. Create a list file for MongoDB:

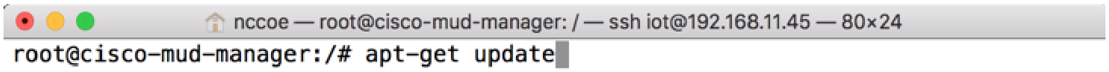
      ```
      echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-
      org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list
      ```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/
ubuntu trusty/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mon
godb-org-4.0.list
```

c. Reload the local package database:

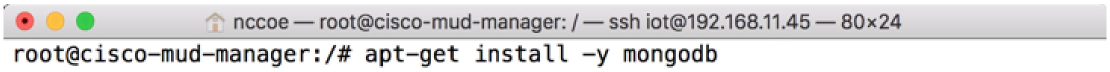
```
apt-get update
```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# apt-get update
```

d. Install the MongoDB packages:

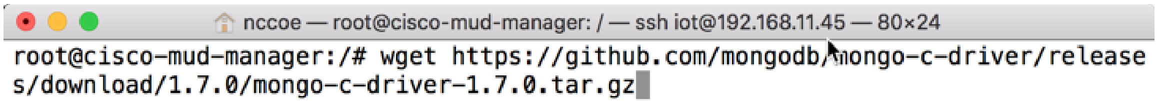
```
apt-get install -y mongodb
```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# apt-get install -y mongodb
```

7. To install the Mongo C driver, enter the following command:

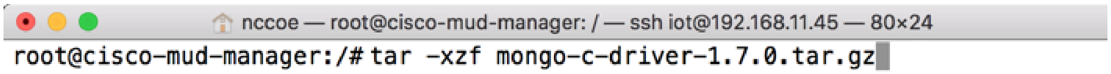
```
wget https://github.com/mongodb/mongo-c-driver/releases/download/1.7.0/mongo-c-driver-1.7.0.tar.gz
```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# wget https://github.com/mongodb/mongo-c-driver/release
s/download/1.7.0/mongo-c-driver-1.7.0.tar.gz
```

a. Untar the file by entering the following command:

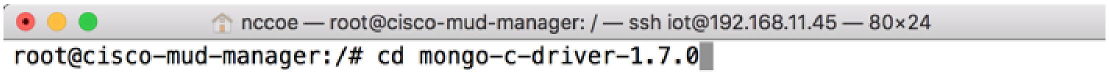
```
tar -xzf mongo-c-driver-1.7.0.tar.gz
```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# tar -xzf mongo-c-driver-1.7.0.tar.gz
```

b. Change into the mongo-c-driver-1.7.0 directory by entering the following command:

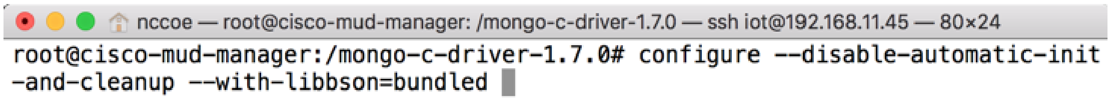
```
cd mongo-c-driver-1.7.0/
```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# cd mongo-c-driver-1.7.0
```

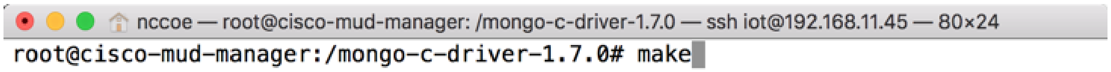
c. Build the Mongo C driver by entering the following commands:

```
./configure --disable-automatic-init-and-cleanup --with-libbson=bundled
```


nccoe — root@cisco-mud-manager: /mongo-c-driver-1.7.0 — ssh iot@192.168.11.45 — 80×24

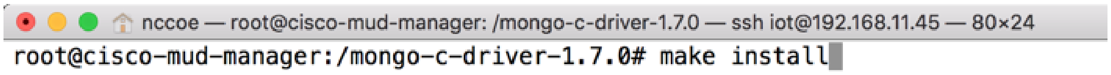
```
root@cisco-mud-manager:/mongo-c-driver-1.7.0# configure --disable-automatic-init
-and-cleanup --with-libbson=bundled
```

```
make
```


nccoe — root@cisco-mud-manager: /mongo-c-driver-1.7.0 — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/mongo-c-driver-1.7.0# make
```

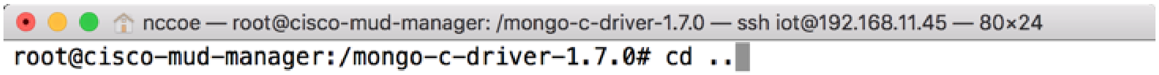
```
make install
```


nccoe — root@cisco-mud-manager: /mongo-c-driver-1.7.0 — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/mongo-c-driver-1.7.0# make install
```

8. Change directories back a folder by entering the following command:

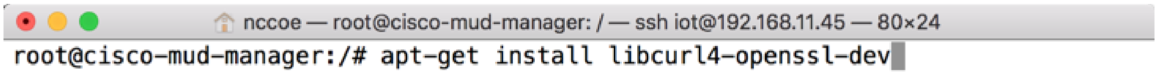
   ```
   cd ..
   ```


nccoe — root@cisco-mud-manager: /mongo-c-driver-1.7.0 — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/mongo-c-driver-1.7.0# cd ..
```

9. To install libcurl, enter the following command:

   ```
   sudo apt-get install libcurl4-openssl-dev
   ```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

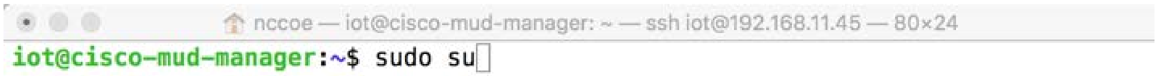
```
root@cisco-mud-manager:/# apt-get install libcurl4-openssl-dev
```

2.1.3.2 MUD Manager Installation

A portion of the steps in this section are documented on Cisco's DevNet GitHub page: https://github.com/CiscoDevNet/MUD-Manager/tree/3.0.1#building-the-mud-manager

1. Open a terminal window, and enter the following command to log in as root:

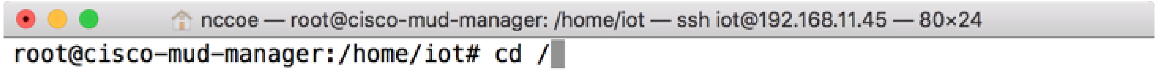
   ```
   sudo su
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo su
```

2. Change to the root directory by entering the following command:

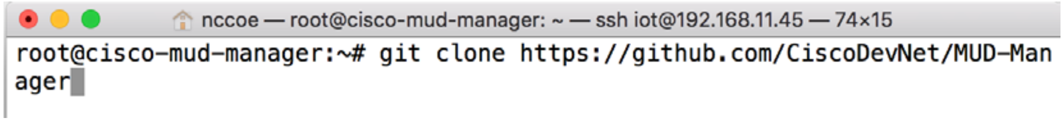
   ```
   cd /
   ```


nccoe — root@cisco-mud-manager: /home/iot — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/home/iot# cd /
```

3. To install the MUD manager, download it from Cisco's GitHub by entering the following command:

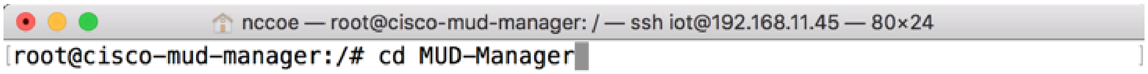
   ```
   git clone https://github.com/CiscoDevNet/MUD-Manager.git
   ```


nccoe — root@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 74×15

```
root@cisco-mud-manager:~# git clone https://github.com/CiscoDevNet/MUD-Man
ager
```

4. Change into the MUD manager directory:

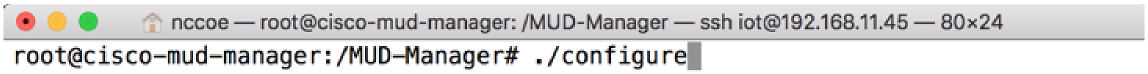
   ```
   cd MUD-Manager
   ```


nccoe — root@cisco-mud-manager: / — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/# cd MUD-Manager
```

5. Build the MUD manager by entering the following commands:

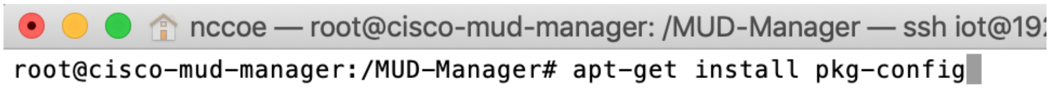
```
./configure
```


nccoe — root@cisco-mud-manager: /MUD-Manager — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/MUD-Manager# ./configure
```

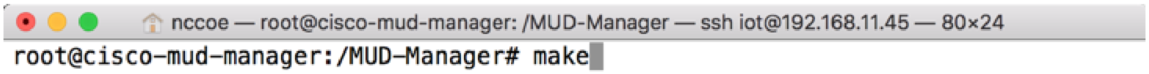
Note: If a “pkg-config error” is thrown, run the command below to install the missing package:

```
apt-get install pkg-config
```



```
root@cisco-mud-manager:/MUD-Manager# apt-get install pkg-config
```

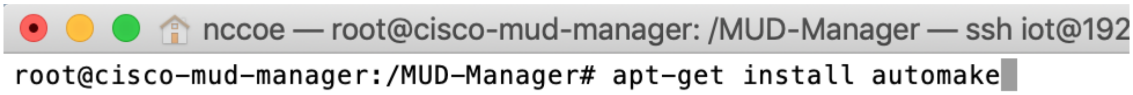
```
make
```


nccoe — root@cisco-mud-manager: /MUD-Manager — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/MUD-Manager# make
```

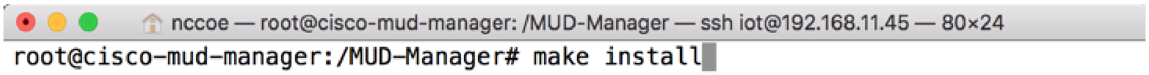
Note: If an “ac.local error” is thrown, run the command below to install the missing package:

```
apt-get install automake
```



```
root@cisco-mud-manager:/MUD-Manager# apt-get install automake
```

```
make install
```


nccoe — root@cisco-mud-manager: /MUD-Manager — ssh iot@192.168.11.45 — 80×24

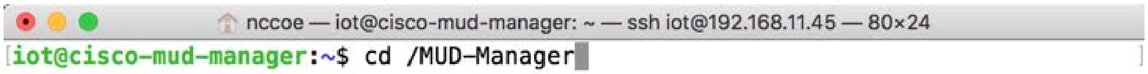
```
root@cisco-mud-manager:/MUD-Manager# make install
```

2.1.3.3 MUD Manager Configuration

This section describes configuring the MUD manager to communicate with the NCCoE MUD file server and defining the attributes used for translating the fetched MUD files. Details about the configuration file and additional fields that can be set within this file can be accessed here: https://github.com/CiscoDevNet/MUD-Manager#editing-the-configuration-file.

1. In the terminal, change to the MUD manager directory:

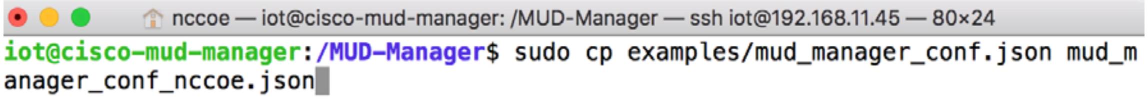
```
cd /MUD-Manager
```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ cd /MUD-Manager
```

2. Copy the contents of the sample *mud_manager_conf.json* file to a different file:

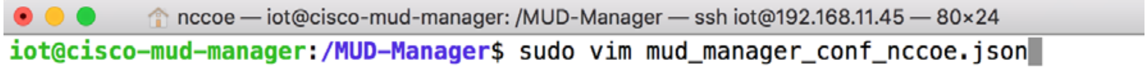
   ```
   sudo cp examples/mud_manager_conf.json mud_manager_conf_nccoe.json
   ```


nccoe — iot@cisco-mud-manager: /MUD-Manager — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:/MUD-Manager$ sudo cp examples/mud_manager_conf.json mud_m
anager_conf_nccoe.json
```

3. Modify the contents of the new MUD manager configuration file:

   ```
   sudo vim mud_manager_conf_nccoe.json
   ```


nccoe — iot@cisco-mud-manager: /MUD-Manager — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:/MUD-Manager$ sudo vim mud_manager_conf_nccoe.json
```

```
{
       "MUD_Manager_Version" : 3,
       "MUDManagerAPIProtocol" : "http",
       "ACL_Prefix" : "ACS:",
       "ACL_Type" : "dACL-ingress-only",
       "COA_Password" : "cisco",
       "VLANs" : [
              {      "VLAN_ID" : 3,
                     "v4addrmask" : "192.168.13.0 0.0.0.255"
              },
              {      "VLAN_ID" : 4,
                     "v4addrmask" : "192.168.14.0 0.0.0.255"
              },
              {      "VLAN_ID" : 5,
                     "v4addrmask" : "192.168.15.0 0.0.0.255"
              }
       ],
       "Manufacturers" : [
              { "authority" : "mudfileserver",
                "cert" : "/home/mudtester/digicertca-chain.crt",
                "web_cert": "/home/mudtester/digicertchain.pem",
                "my_controller_v4" : "192.168.10.125",
                "my_controller_v6" : "2610:20:60CE:630:B000::7",
                "local_networks_v4" : "192.168.10.0 0.0.0.255",
                "local_networks_v6" : "2610:20:60CE:630:B000::",
                "vlan_nw_v4" : "192.168.13.0 0.0.0.255",
                "vlan" : 3
              },
              {
              "authority" : "www.gmail.com",
                "cert" : "/home/mudtester/digicertca-chain.crt",
                "web_cert": "/home/mudtester/digicertchain.pem",
                "vlan_nw_v4" : "192.168.14.0 0.0.0.255",
                "vlan" : 4
```

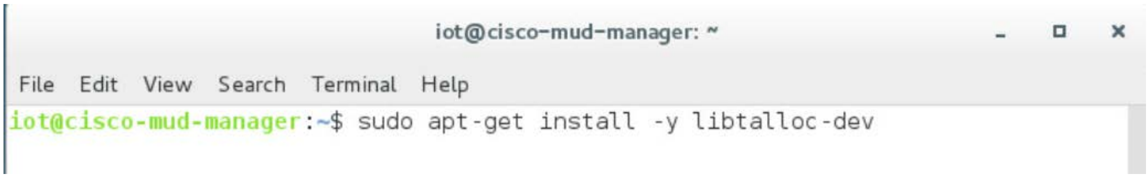
```
                }
        ],
        "DNSMapping" : {
                "www.osmud.org" : "198.71.233.87",
                "www.mqttbroker.com" : "192.168.4.6",
                "us.dlink.com" : "54.187.217.118",
                "www.nossl.net": "40.68.201.127",
                "www.trytechy.com" : "99.84.104.21"
        },

        "DNSMapping_v6" : {
                "www.mqttbroker.com" : "2610:20:60CE:630:B000::6",
                "www.updateserver.com" : "2610:20:60CE:630:B000::7",
                "www.dominiontea.com": "2a03:2880:f10c:83:face:b00c:0:25de"
        },
        "ControllerMapping" : {
                "https://www.google.com" : "192.168.10.104",
                "http://lightcontroller.example2.com": "192.168.4.77",
                "http://lightcontroller.example.com": "192.168.4.78"
        },
        "ControllerMapping_v6" : {
                "https:/www.google.com" : "ffff:2343:4444:::",
                "http://lightcontroller.example2.com": "ffff:2343:4444:::",
                "http://lightcontroller.example.com": "ffff:2343:4444:::"

        },
        "DefaultACL" : ["permit tcp any eq 22 any","permit udp any eq 68 any eq
67","permit udp any any eq 53", "deny ip any any"],
        "DefaultACL_v6" : ["permit udp any any eq 53", "deny ipv6 any any"]

}
```

Details about the contents of the configuration file can be found at the link provided at the start of this section.

2.1.3.4 FreeRADIUS Installation

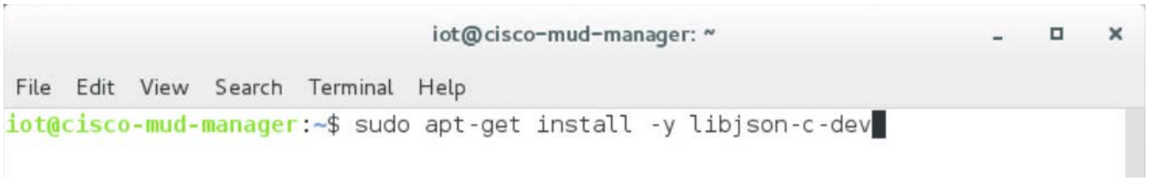
1. Install the dependencies for FreeRADIUS:

   a. `sudo apt-get install -y libtalloc-dev`



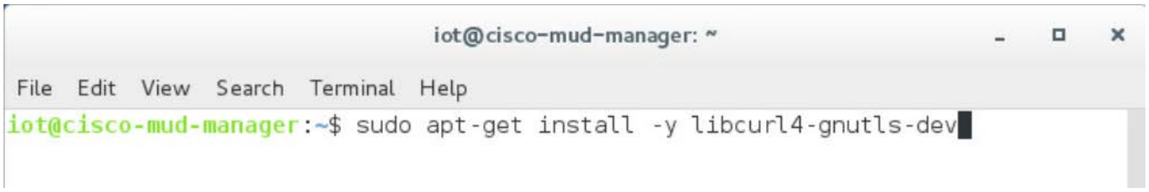
```
iot@cisco-mud-manager:~$ sudo apt-get install -y libtalloc-dev
```

   b. `sudo apt-get install -y libjson-c-dev`



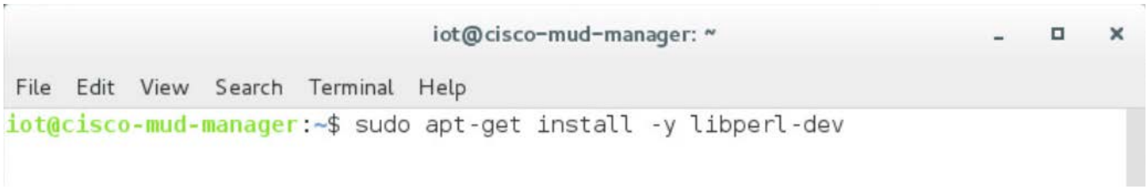
```
iot@cisco-mud-manager:~$ sudo apt-get install -y libjson-c-dev
```

c. `sudo apt-get install -y libcurl4-gnutls-dev`



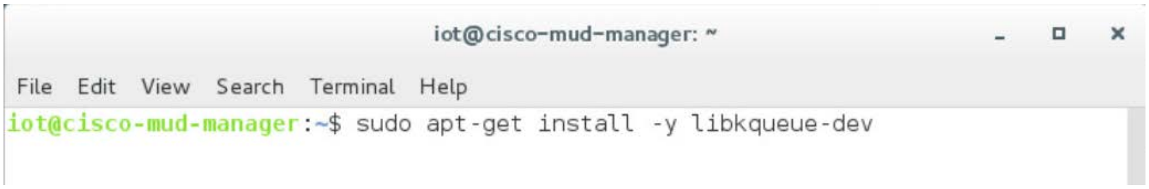
```
iot@cisco-mud-manager:~$ sudo apt-get install -y libcurl4-gnutls-dev
```

d. `sudo apt-get install -y libperl-dev`



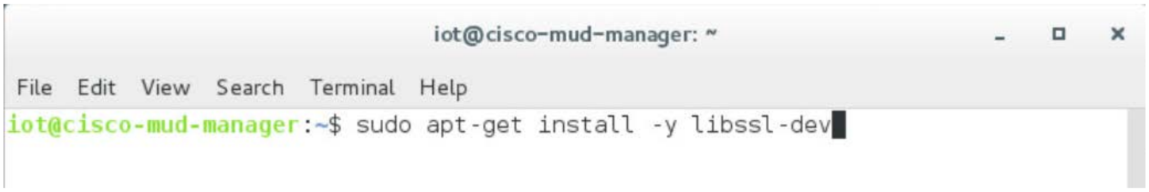
```
iot@cisco-mud-manager:~$ sudo apt-get install -y libperl-dev
```

e. `sudo apt-get install -y libkqueue-dev`



```
iot@cisco-mud-manager:~$ sudo apt-get install -y libkqueue-dev
```

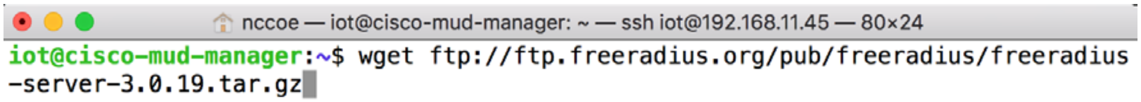
f. `sudo apt-get install -y libssl-dev`



```
iot@cisco-mud-manager:~$ sudo apt-get install -y libssl-dev
```

2. Download the source by entering the following command (Note: Version 3.0.19 and later are recommended):

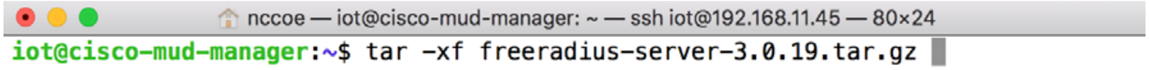
   `wget` ftp://ftp.freeradius.org/pub/freeradius/freeradius-server-3.0.19.tar.gz


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ wget ftp://ftp.freeradius.org/pub/freeradius/freeradius
-server-3.0.19.tar.gz
```

3. Untar the downloaded file by entering the following command:

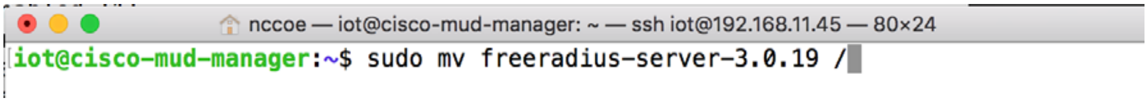
```
tar -xf freeradius-server-3.0.19.tar.gz
```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ tar -xf freeradius-server-3.0.19.tar.gz
```

4. Move the FreeRADIUS directory to the root directory:

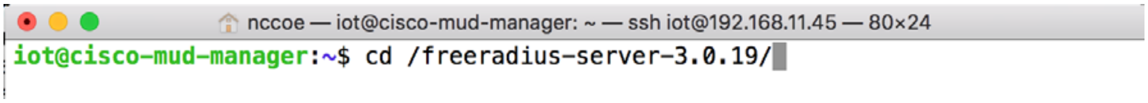
```
sudo mv freeradius-server-3.0.19/ /
```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo mv freeradius-server-3.0.19 /
```

5. Change to the FreeRADIUS directory:

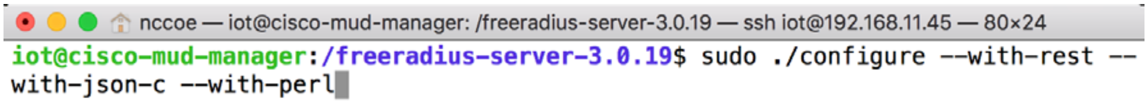
```
cd /freeradius-server-3.0.19/
```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ cd /freeradius-server-3.0.19/
```

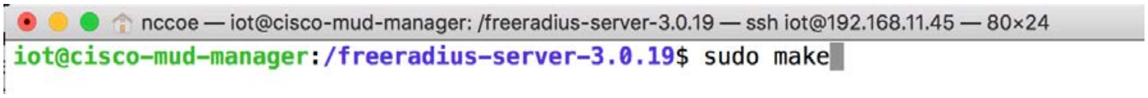
6. Make and install the source by entering the following:

   a. `sudo ./configure --with-rest --with-json-c --with-perl`


nccoe — iot@cisco-mud-manager: /freeradius-server-3.0.19 — ssh iot@192.168.11.45 — 80×24

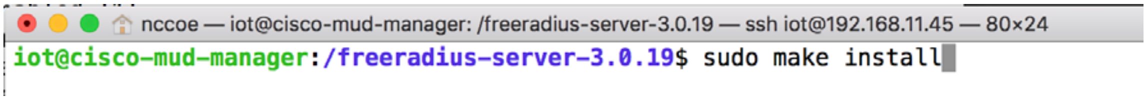
```
iot@cisco-mud-manager:/freeradius-server-3.0.19$ sudo ./configure --with-rest --
with-json-c --with-perl
```

   b. `sudo make`


nccoe — iot@cisco-mud-manager: /freeradius-server-3.0.19 — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:/freeradius-server-3.0.19$ sudo make
```

   c. `sudo make install`

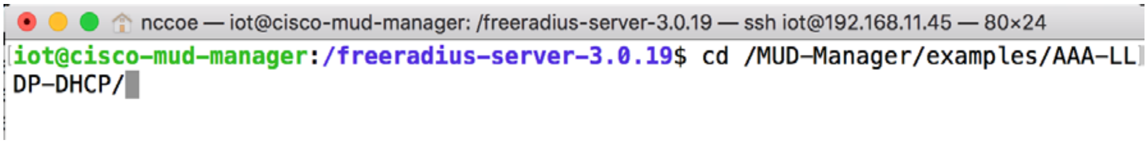

nccoe — iot@cisco-mud-manager: /freeradius-server-3.0.19 — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:/freeradius-server-3.0.19$ sudo make install
```

2.1.3.5 FreeRADIUS Configuration

1. Change to the FreeRADIUS subdirectory in the MUD manager directory:

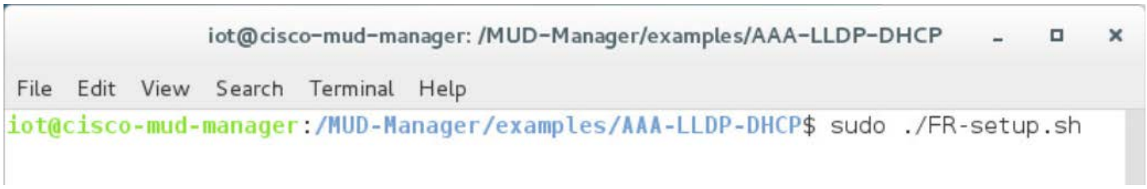
```
cd /MUD-Manager/examples/AAA-LLDP-DHCP/
```


nccoe — iot@cisco-mud-manager: /freeradius-server-3.0.19 — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:/freeradius-server-3.0.19$ cd /MUD-Manager/examples/AAA-LL
DP-DHCP/
```

2. Run the setup script:

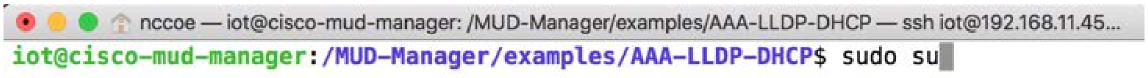
```
sudo ./FR-setup.sh
```


iot@cisco-mud-manager: /MUD-Manager/examples/AAA-LLDP-DHCP
File Edit View Search Terminal Help

```
iot@cisco-mud-manager:/MUD-Manager/examples/AAA-LLDP-DHCP$ sudo ./FR-setup.sh
```

3. Enter the following command to log in as root:

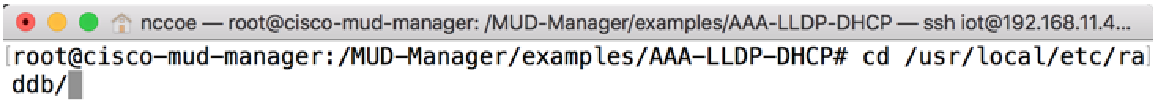
```
sudo su
```


nccoe — iot@cisco-mud-manager: /MUD-Manager/examples/AAA-LLDP-DHCP — ssh iot@192.168.11.45...

```
iot@cisco-mud-manager:/MUD-Manager/examples/AAA-LLDP-DHCP$ sudo su
```

4. Change to the RADIUS directory:

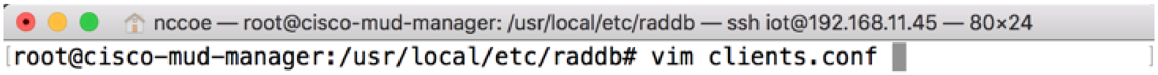
```
cd /usr/local/etc/raddb/
```


nccoe — root@cisco-mud-manager: /MUD-Manager/examples/AAA-LLDP-DHCP — ssh iot@192.168.11.4...

```
root@cisco-mud-manager:/MUD-Manager/examples/AAA-LLDP-DHCP# cd /usr/local/etc/ra
ddb/
```

5. Open the *clients.conf* file:

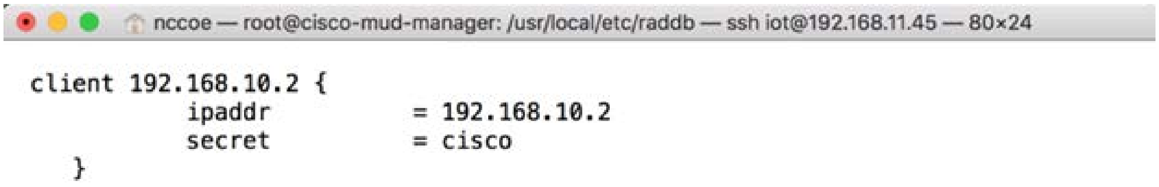
```
vim clients.conf
```


nccoe — root@cisco-mud-manager: /usr/local/etc/raddb — ssh iot@192.168.11.45 — 80×24

```
root@cisco-mud-manager:/usr/local/etc/raddb# vim clients.conf
```

6. Add the network access server (NAS) as an authorized client in the configuration file on the server by adding an entry for the NAS in the *client.conf* file that is opened (Note: Replace the IP address below with the IP address of the NAS, and insert the "secret" configured on the NAS to talk to the RADIUS servers):

```
client 192.168.10.2 {
      ipaddr = 192.168.10.2
      secret = cisco
   }
```


nccoe — root@cisco-mud-manager: /usr/local/etc/raddb — ssh iot@192.168.11.45 — 80×24

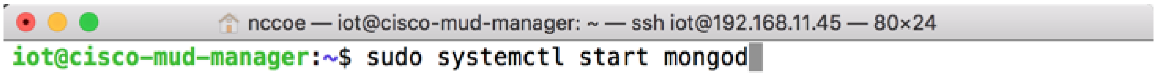
```
client 192.168.10.2 {
          ipaddr          = 192.168.10.2
          secret          = cisco
   }
```

7. Save and close the file.

2.1.3.6 Start MUD Manager and FreeRADIUS Server

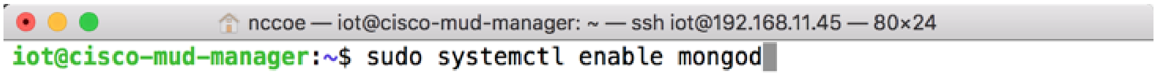
1. Start and enable the database by executing the following commands:

   ```
   sudo systemctl start mongod
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo systemctl start mongod
```

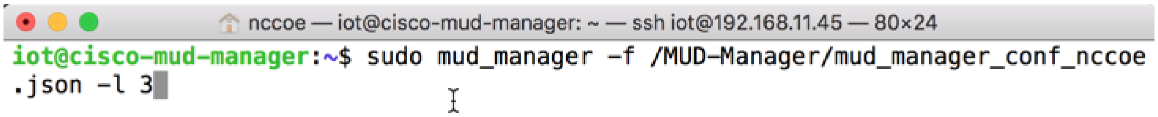
   ```
   sudo systemctl enable mongod
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo systemctl enable mongod
```

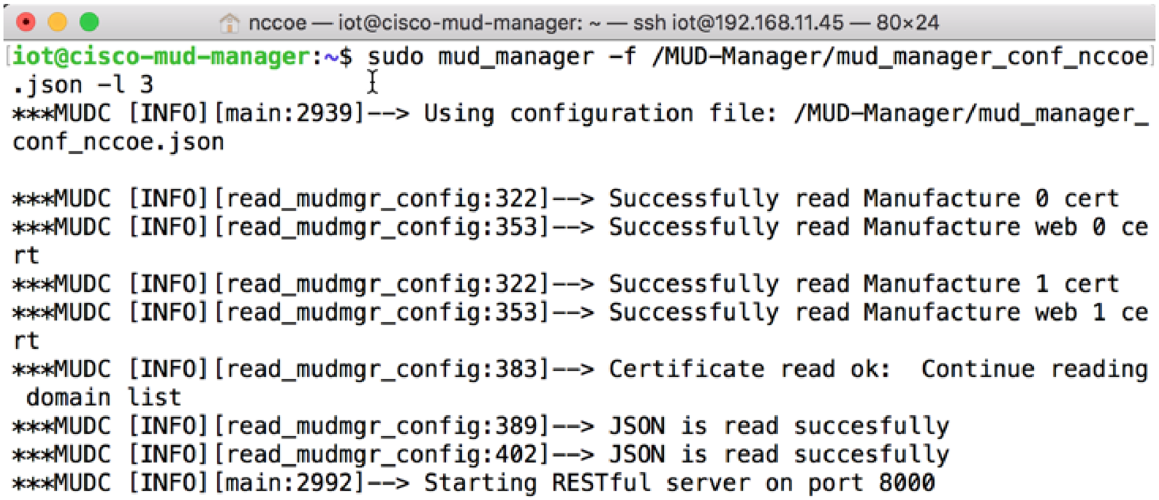
2. Start the MUD manager in the foreground with logging enabled by entering the following command:

   ```
   sudo mud_manager -f /MUD-Manager/mud_manager_conf_nccoe.json -l 3
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo mud_manager -f /MUD-Manager/mud_manager_conf_nccoe
.json -l 3
```

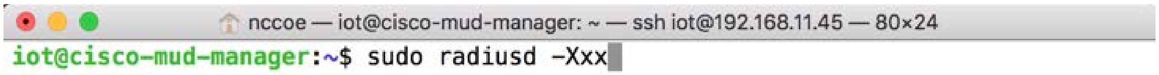
   The following output should appear if the service started successfully:


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo mud_manager -f /MUD-Manager/mud_manager_conf_nccoe
.json -l 3
***MUDC [INFO][main:2939]--> Using configuration file: /MUD-Manager/mud_manager_
conf_nccoe.json

***MUDC [INFO][read_mudmgr_config:322]--> Successfully read Manufacture 0 cert
***MUDC [INFO][read_mudmgr_config:353]--> Successfully read Manufacture web 0 ce
rt
***MUDC [INFO][read_mudmgr_config:322]--> Successfully read Manufacture 1 cert
***MUDC [INFO][read_mudmgr_config:353]--> Successfully read Manufacture web 1 ce
rt
***MUDC [INFO][read_mudmgr_config:383]--> Certificate read ok:  Continue reading
 domain list
***MUDC [INFO][read_mudmgr_config:389]--> JSON is read succesfully
***MUDC [INFO][read_mudmgr_config:402]--> JSON is read succesfully
***MUDC [INFO][main:2992]--> Starting RESTful server on port 8000
```

3. Start the FreeRADIUS service in the foreground with logging enabled by entering the following command:

   ```
   sudo radiusd -Xxx
   ```


nccoe — iot@cisco-mud-manager: ~ — ssh iot@192.168.11.45 — 80×24

```
iot@cisco-mud-manager:~$ sudo radiusd -Xxx
```

At this point all the processes required to support MUD are running on the server side, and the next step is to configure the Cisco Catalyst switch. Once the switch configuration detailed in the Cisco Switch–Catalyst 3850-S setup section is completed, any DHCP activity on the network should appear in the output of the FreeRADIUS and MUD manager logs.

2.2 MUD File Server

2.2.1 MUD File Server Overview

For this build, the NCCoE built a MUD file server hosted within the lab infrastructure. This file server signs and stores the MUD files along with their corresponding signature files for the MUD-capable IoT devices used in the build. The MUD file server is also responsible for serving the MUD file and the corresponding signature file upon request from the MUD manager.

2.2.2 Configuration Overview

The following subsections document the software and network configurations for the MUD file server.

2.2.2.1 Network Configuration

This server was hosted in the NCCoE's virtual environment, functioning as a cloud service. Its IP address was statically assigned.

2.2.2.2 Software Configuration

For this build, the server ran on the CentOS 7 operating system. The MUD files and signatures were hosted by an Apache web server and configured to use secure sockets layer/Transport Layer Security (SSL/TLS) encryption.

2.2.2.3 Hardware Configuration

The MUD file server was hosted in the NCCoE's virtual environment, functioning as a cloud service.

2.2.3 Setup

The following subsections describe the process for configuring the MUD file server.

2.2.3.1 Apache Web Server

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. After that, SSL/TLS encryption was set up by using the digital certificate and key obtained from DigiCert. This was set up by using the official Apache documentation, found at https://httpd.apache.org/docs/current/ssl/ssl_howto.html.

2.2.3.2 MUD File Creation and Signing

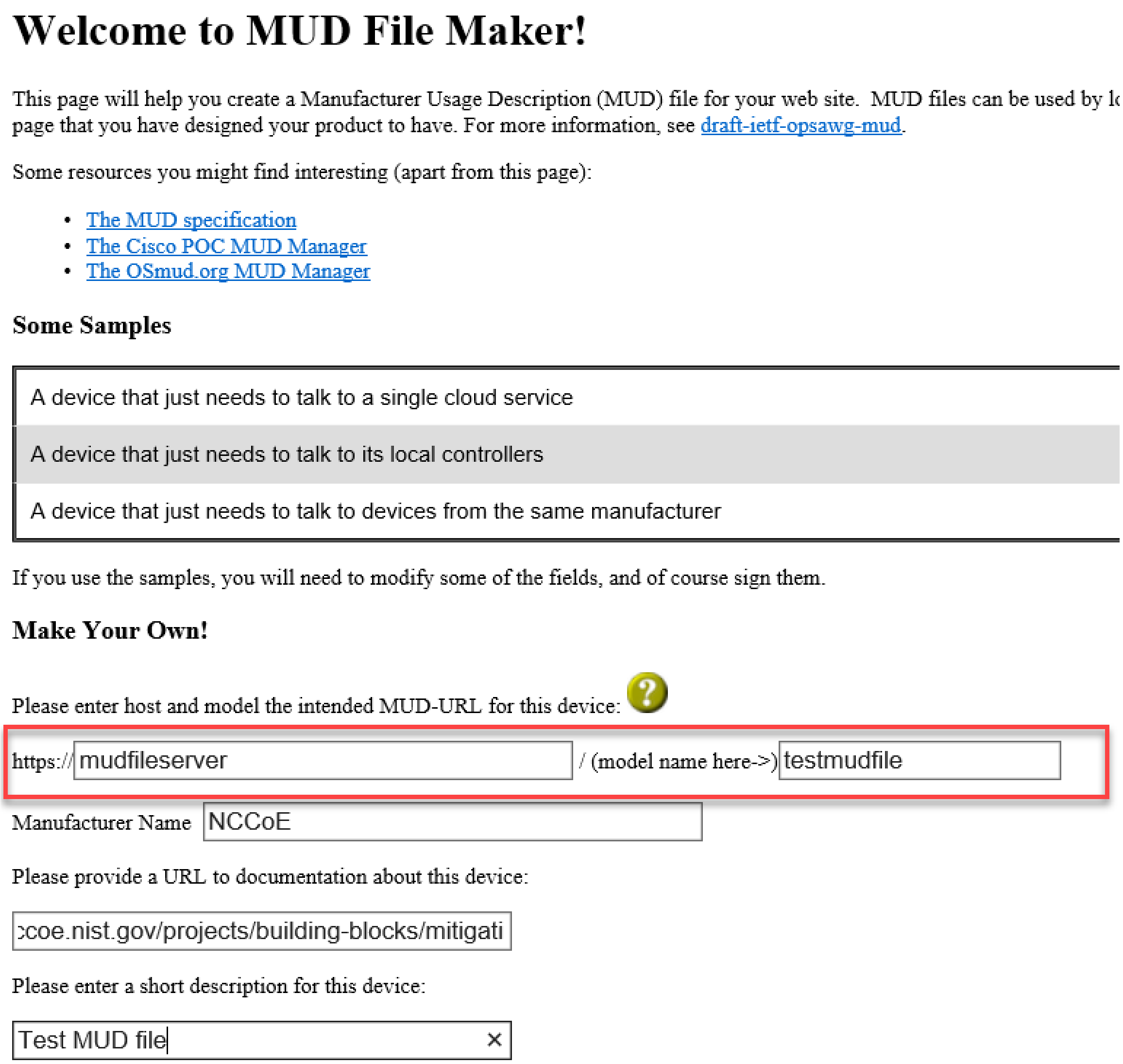
This section details creating and signing a MUD file on the MUD file server. The MUD specification does not mandate that this signing process be performed on the MUD file server itself.

2.2.3.2.1 MUD File Creation

An online tool called MUD Maker was used to build MUD files. Once the permitted communications have been defined for the IoT device, proceed to www.mudmaker.org to leverage the online tool. There is also a list of sample MUD files on the site, which can be used as a reference. Upon navigating to www.mudmaker.org, complete the following steps to create a MUD file:

1. Specify the host that will be serving the MUD file and the model name of the device in the appropriate input fields, which are outlined in red in the screenshot below (Note: This will result in the MUD URL for this device):

   Sample input: mudfileserver, testmudfile



Welcome to MUD File Maker!

This page will help you create a Manufacturer Usage Description (MUD) file for your web site. MUD files can be used by lo
page that you have designed your product to have. For more information, see draft-ietf-opsawg-mud.

Some resources you might find interesting (apart from this page):

- The MUD specification
- The Cisco POC MUD Manager
- The OSmud.org MUD Manager

Some Samples

A device that just needs to talk to a single cloud service
A device that just needs to talk to its local controllers
A device that just needs to talk to devices from the same manufacturer

If you use the samples, you will need to modify some of the fields, and of course sign them.

Make Your Own!

Please enter host and model the intended MUD-URL for this device: ?

https:// mudfileserver / (model name here->) testmudfile

Manufacturer Name NCCoE

Please provide a URL to documentation about this device:

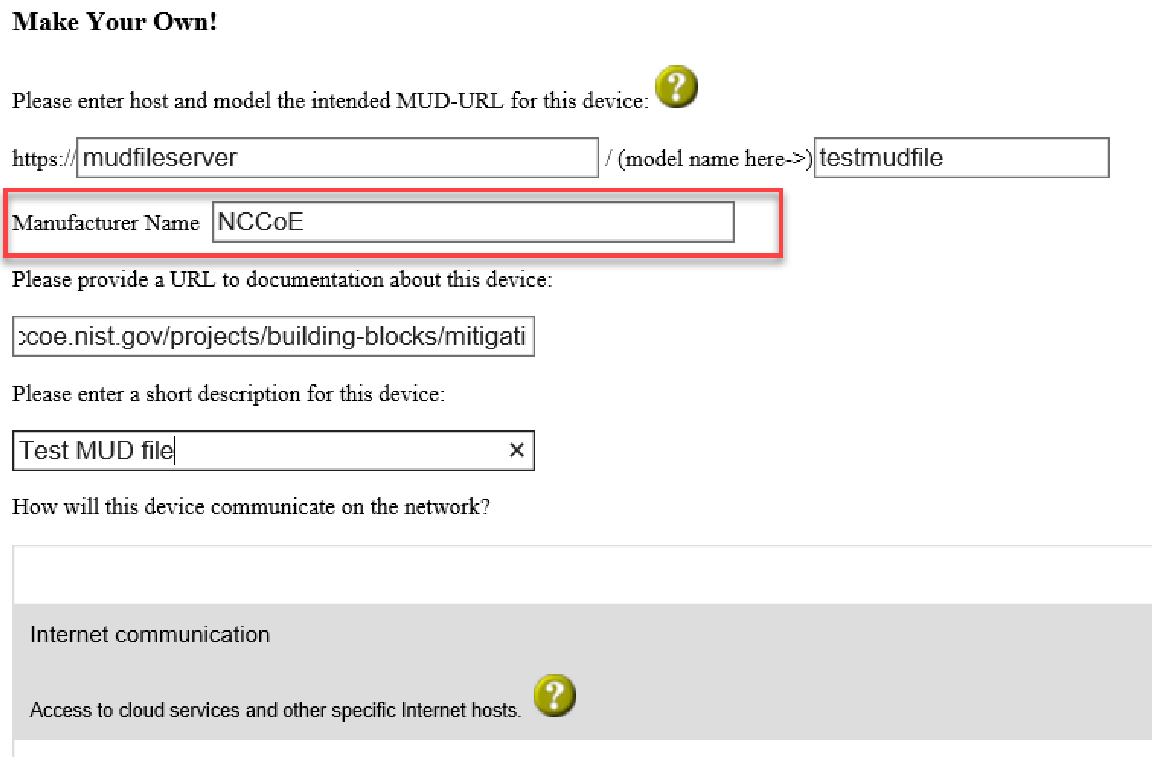
:coe.nist.gov/projects/building-blocks/mitigati

Please enter a short description for this device:

Test MUD file ×

786

787 2. Specify the Manufacturer Name of the device in the appropriate input field, which is outlined in
788 red in the screenshot below:



Make Your Own!

Please enter host and model the intended MUD-URL for this device:

https:// mudfileserver / (model name here->) testmudfile

Manufacturer Name NCCoE

Please provide a URL to documentation about this device:

:coe.nist.gov/projects/building-blocks/mitigati

Please enter a short description for this device:

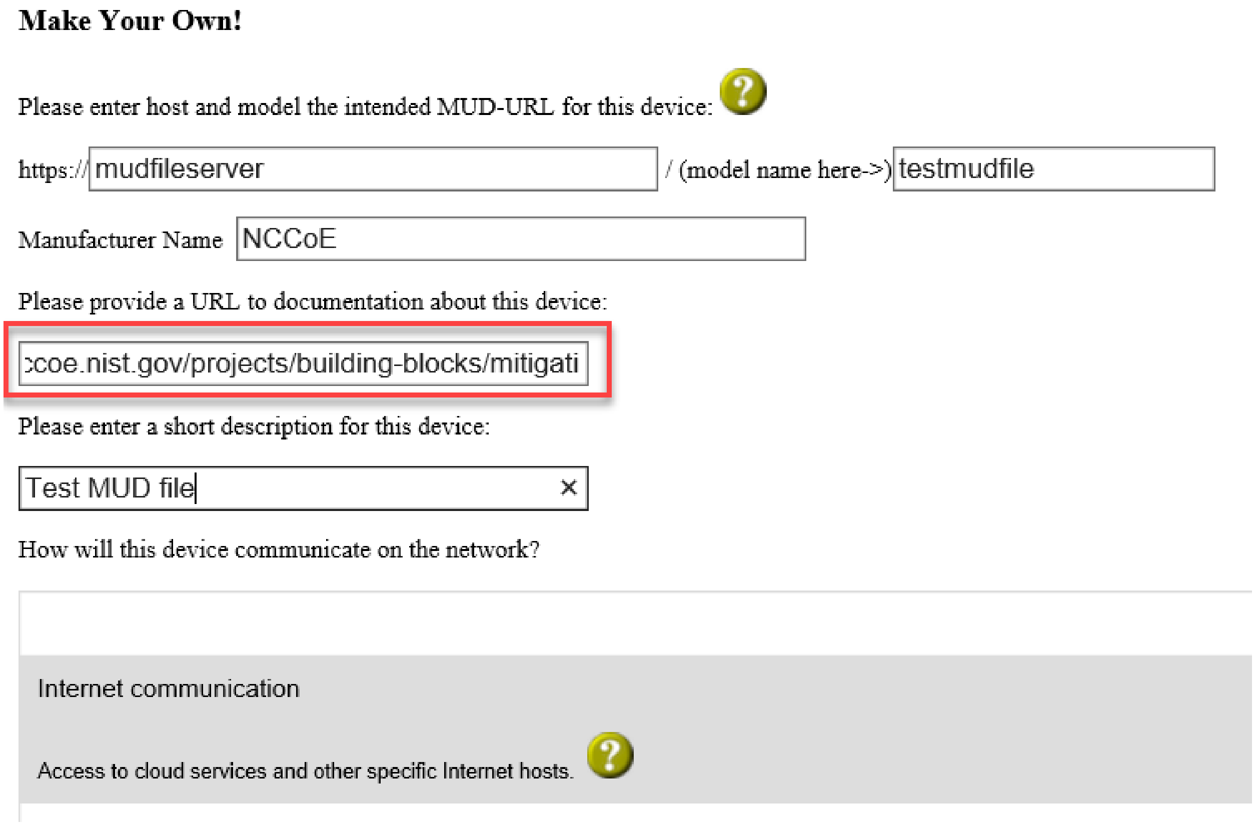
Test MUD file

How will this device communicate on the network?

Internet communication

Access to cloud services and other specific Internet hosts.

3. Include a URL to provide documentation about this device in the appropriate input field, which is outlined in red in the screenshot below:



Make Your Own!

Please enter host and model the intended MUD-URL for this device:

https:// mudfileserver / (model name here->) testmudfile

Manufacturer Name NCCoE

Please provide a URL to documentation about this device:

:coe.nist.gov/projects/building-blocks/mitigati

Please enter a short description for this device:

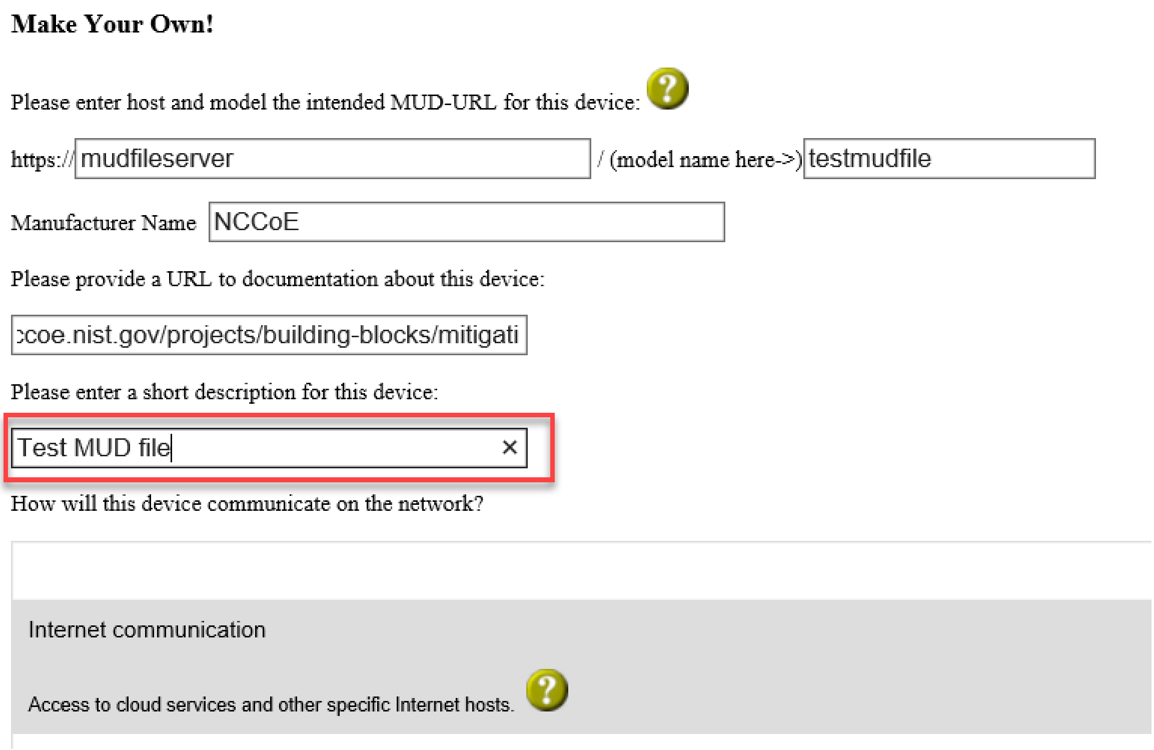
Test MUD file

How will this device communicate on the network?

Internet communication

Access to cloud services and other specific Internet hosts.

4. Include a short description of the device in the appropriate input field, which is outlined in red in the screenshot below:



Make Your Own!

Please enter host and model the intended MUD-URL for this device:

https:// mudfileserver / (model name here->) testmudfile

Manufacturer Name NCCoE

Please provide a URL to documentation about this device:

:coe.nist.gov/projects/building-blocks/mitigati

Please enter a short description for this device:

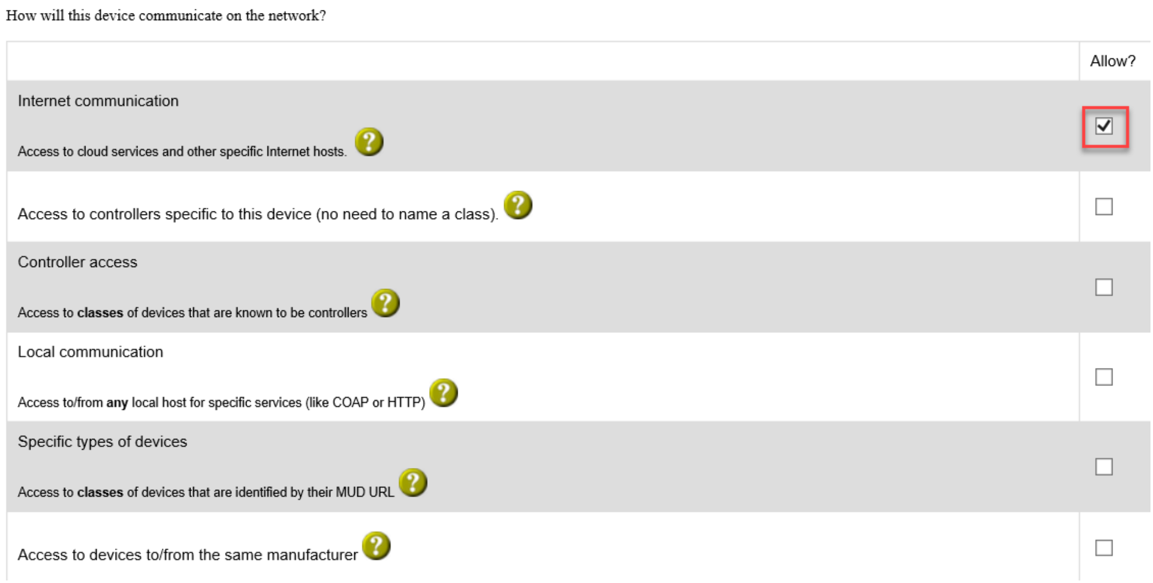
Test MUD file

How will this device communicate on the network?

Internet communication

Access to cloud services and other specific Internet hosts.

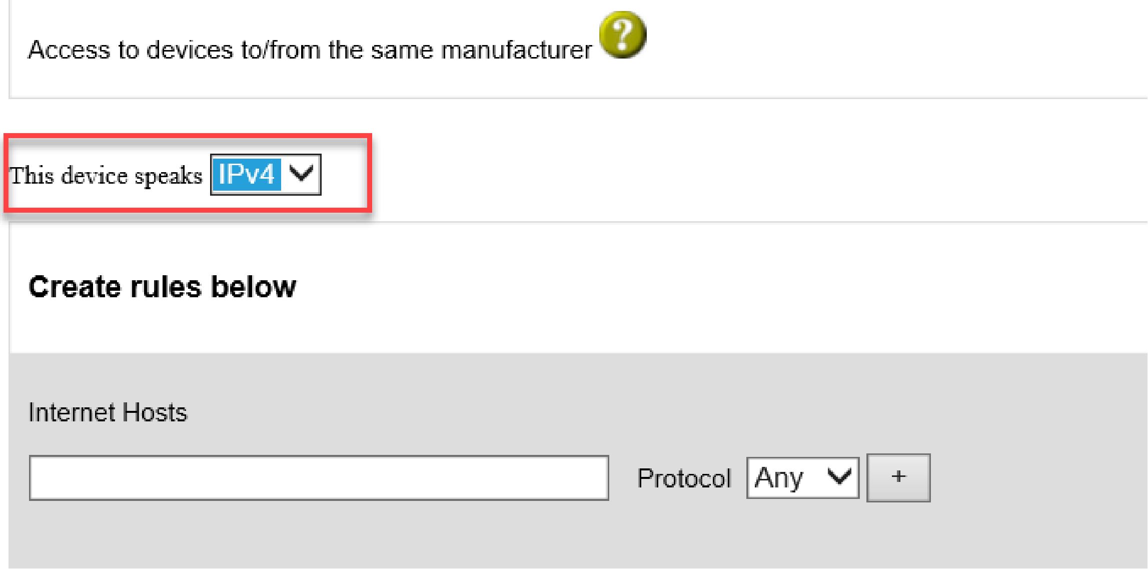
5. Check the boxes for the types of network communication that are allowed for the device:



How will this device communicate on the network?

	Allow?
Internet communication Access to cloud services and other specific Internet hosts.	☑
Access to controllers specific to this device (no need to name a class).	□
Controller access Access to **classes** of devices that are known to be controllers	□
Local communication Access to/from **any** local host for specific services (like COAP or HTTP)	□
Specific types of devices Access to **classes** of devices that are identified by their MUD URL	□
Access to devices to/from the same manufacturer	□

6. Specify the internet protocol version that the device leverages:



Access to devices to/from the same manufacturer

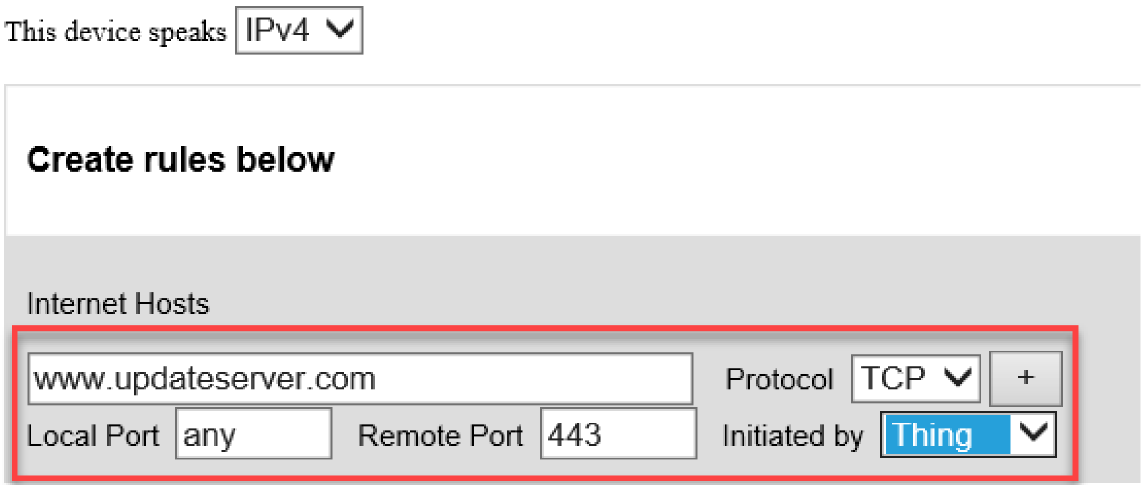
This device speaks IPv4

Create rules below

Internet Hosts

Protocol Any

7. Specify values for the fields (Internet Hosts, Protocol, Local Port, Remote Port, and Initiated by) that describe the communications that will be permitted for the device:



This device speaks IPv4

Create rules below

Internet Hosts

www.updateserver.com

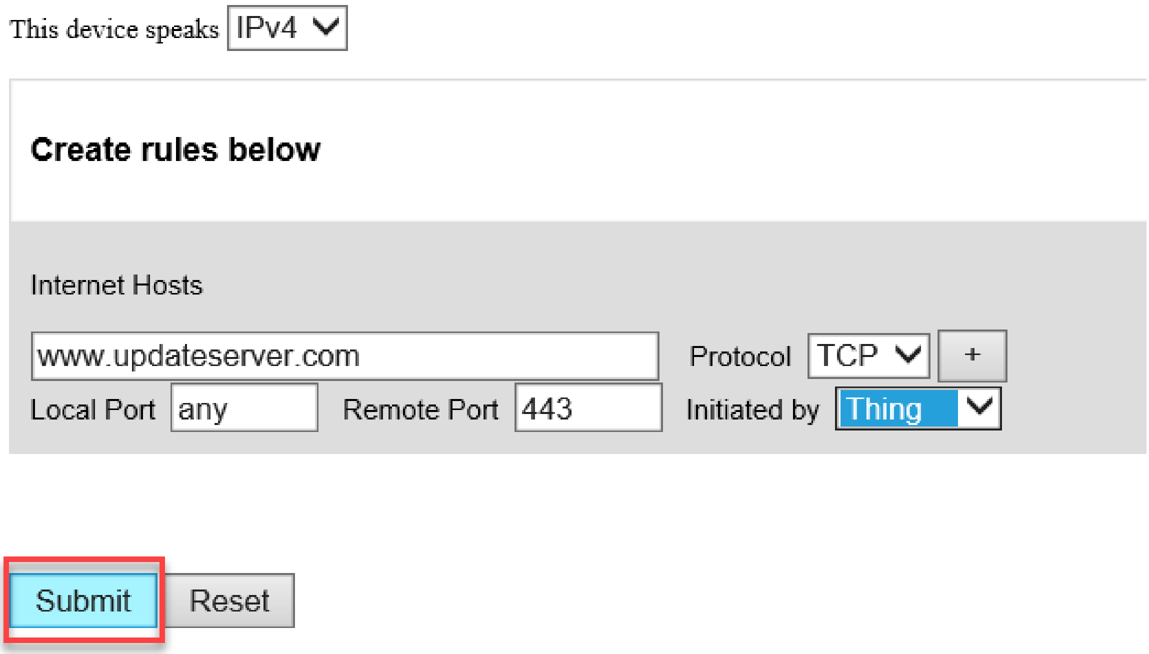
Protocol TCP

Local Port any

Remote Port 443

Initiated by Thing

8. Click **Submit** to generate the MUD file:



This device speaks: IPv4

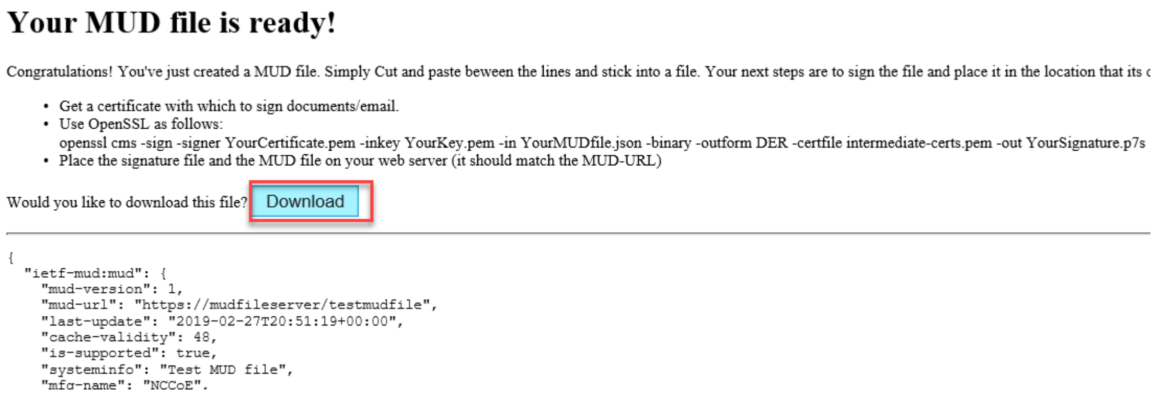
Create rules below

Internet Hosts

www.updateserver.com Protocol: TCP

Local Port: any Remote Port: 443 Initiated by: Thing

9. Once completed, the page will redirect to the following page that outputs the MUD file on the screen. Click **Download** to download the MUD file, which is a .JSON file:



Your MUD file is ready!

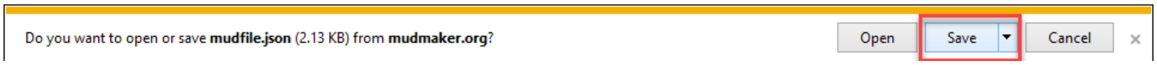
Congratulations! You've just created a MUD file. Simply Cut and paste beween the lines and stick into a file. Your next steps are to sign the file and place it in the location that its

- Get a certificate with which to sign documents/email.
- Use OpenSSL as follows:
  openssl cms -sign -signer YourCertificate.pem -inkey YourKey.pem -in YourMUDfile.json -binary -outform DER -certfile intermediate-certs.pem -out YourSignature.p7s
- Place the signature file and the MUD file on your web server (it should match the MUD-URL)

Would you like to download this file?

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://mudfileserver/testmudfile",
    "last-update": "2019-02-27T20:51:19+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "Test MUD file",
    "mfg-name": "NCCoE",
```

10. Click **Save** to store a copy of the MUD file:



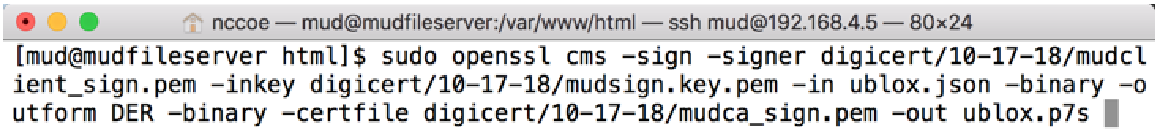
Do you want to open or save **mudfile.json** (2.13 KB) from **mudmaker.org**?

2.2.3.2.2 MUD File Signature Creation and Verification

In this build, OpenSSL is used to sign and verify MUD files. This example uses the MUD file created in the previous section, which is named *ublox.json*; the Signing Certificate; the Private Key for the Signing Certificate; the Intermediate Certificate for the Signing Certificate; and the Certificate of the Trusted Root Certificate Authority (CA) for the Signing Certificate.

1. Sign the MUD file by using the following command:

```
sudo openssl cms -sign -signer <Signing Certificate> -inkey <Private Key for
Signing Certificate> -in <Name of MUD File> -binary -outform DER -binary -
certfile <Intermediate Certificate for Signing Certificate> -out <Name of MUD
File without the .json file extension>.p7s
```

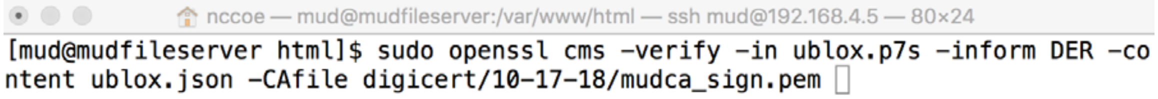

nccoe — mud@mudfileserver:/var/www/html — ssh mud@192.168.4.5 — 80×24

```
[mud@mudfileserver html]$ sudo openssl cms -sign -signer digicert/10-17-18/mudcl
ient_sign.pem -inkey digicert/10-17-18/mudsign.key.pem -in ublox.json -binary -o
utform DER -binary -certfile digicert/10-17-18/mudca_sign.pem -out ublox.p7s
```

This will create a signature file for the MUD file that has the same name as the MUD file but ends with the .p7s file extension, i.e., in our case *ublox.p7s*.

2. Manually verify the MUD file signature by using the following command:

```
sudo openssl cms -verify -in <Name of MUD File>.p7s -inform DER -content <Name
of MUD File>.json -CAfile <Certificate of Trusted Root Certificate Authority
for Signing Certificate>
```


nccoe — mud@mudfileserver:/var/www/html — ssh mud@192.168.4.5 — 80×24

```
[mud@mudfileserver html]$ sudo openssl cms -verify -in ublox.p7s -inform DER -co
ntent ublox.json -CAfile digicert/10-17-18/mudca_sign.pem
```

If a valid file signature was created successfully, a corresponding message should appear. Both the MUD file and MUD file signature should be placed on the MUD file server in the Apache server directory.

2.3 Cisco Switch–Catalyst 3850-S

2.3.1 Cisco 3850-S Catalyst Switch Overview

The switch used in this build is an enterprise-class, layer 3 switch. It is a Cisco Catalyst 3850-S that had been modified to support MUD functionality as a proof-of-concept implementation. In addition to providing DHCP services, the switch acts as a broker for connected IoT devices for authentication, authorization, and accounting through a FreeRADIUS server. The Link Layer Discovery Protocol (LLDP) is enabled on ports that MUD-capable devices are plugged into to help facilitate recognition of connected IoT device features, capabilities, and neighbor relationships at layer 2. Additionally, an access session policy is configured on the switch to enable port control for multihost authentication and port monitoring. The combined effect of these switch configurations is a dynamic access list, which has been generated by the MUD manager, being active on the switch to permit or deny access to and from MUD-capable IoT devices.

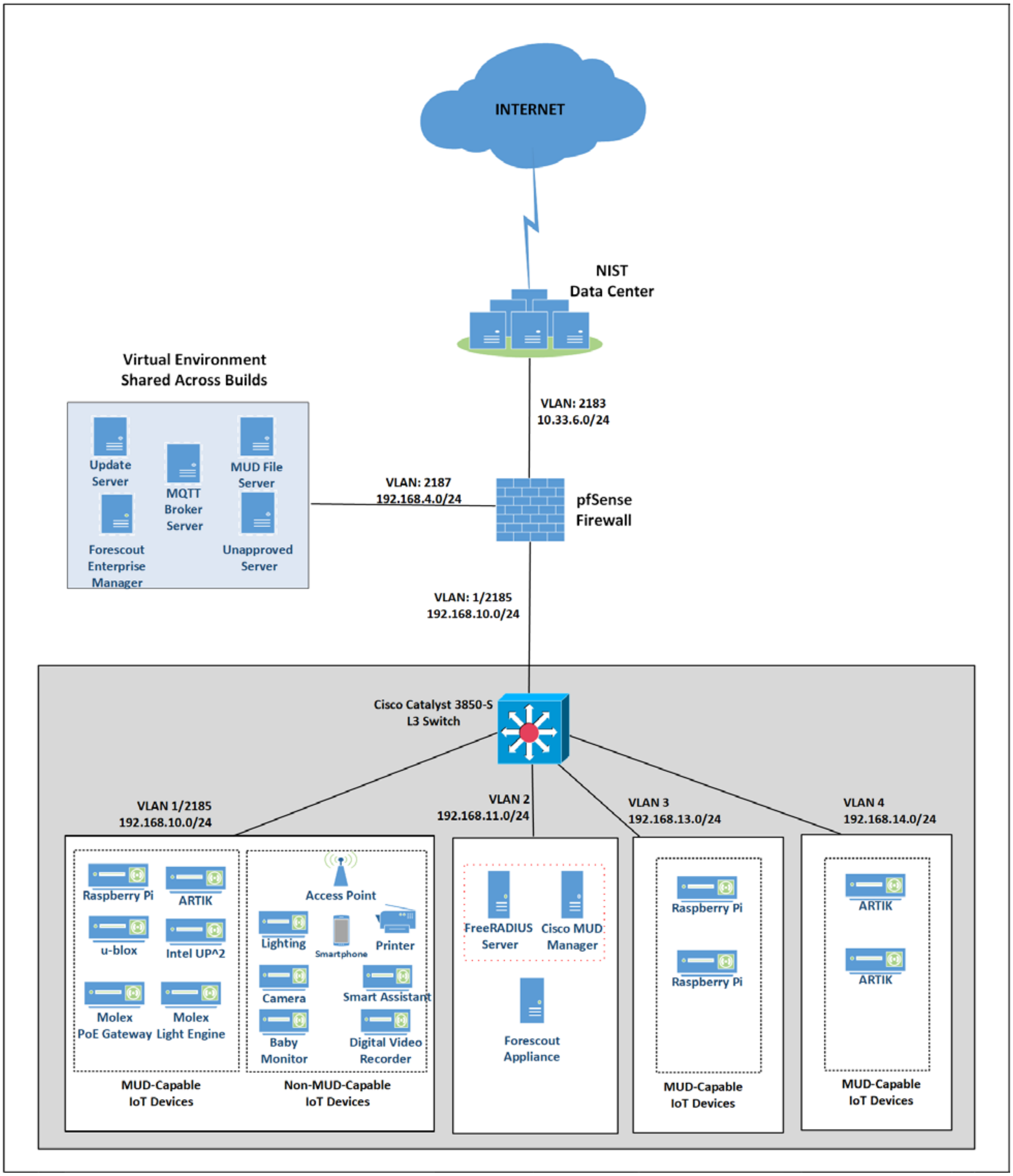
2.3.2 Configuration Overview

The following subsections document the network, software, and hardware configurations for the Cisco Catalyst 3850-S switch.

2.3.2.1 Network Configuration

This section describes how to configure the required Cisco Catalyst 3850-S switch to support the build. A special image for the Catalyst 3850-S was provided by Cisco to support MUD-specific functionality. In our build, the switch is integrated with a DHCP server and a FreeRADIUS server, which together support delivery of the MUD URL to the MUD manager via either DHCP or LLDP. The MUD manager is also able to generate and send a dynamic access list to the switch, via the RADIUS server, to permit or deny access to and from the IoT devices. In addition to hosting directly connected IoT devices on VLANs 1, 3, and 4, the switch hosts both the MUD manager and the FreeRADIUS servers on VLAN 2. As illustrated in Figure 2-1, each locally configured VLAN is protected by a firewall that connects the lab environment to the NIST data center, which provides internet access for all connected devices.

850 **Figure 2-1 Physical Architecture–Build 1**



2.3.2.2 Software Configuration

The prototype, MUD-capable Cisco 3850-S used in this build is running internetwork operating system (IOS) version 16.09.02.

2.3.2.3 Hardware Configuration

The Catalyst 3850-S switch configured in the lab consists of 24 one-gigabit Ethernet ports with two optional 10-gigabit Ethernet uplink ports. A customized version of Cat-OS is installed on the switch. The versions of the OS are as follows:

- Cat3k_caa-guestshell.16
- Cat3k_caa-rpbase.16.06
- Cat3k_caa-rpcore.16.06
- Cat3k_caa-srdriver.16.06.0
- Cat3k_caa-webui.16.06.0

2.3.3 Setup

Table 2-1 lists the Cisco 3850-S switch running configuration used for the lab environment. In addition to the IOS version and a few generic configuration items, configuration items specifically relating to integration with the MUD manager and IoT devices are highlighted in bold fonts; these include DHCP, LLDP, AAA, RADIUS, and policies regarding access session. Table 2-1 also provides a description of each configuration item for ease of understanding.

Table 2-1 Cisco 3850-S Switch Running Configuration

Configuration Item	Description
version 16.9 no service pad service timestamps debug datetime msec service timestamps log datetime msec service call-home no platform punt-keepalive disable-kernel-core ! hostname Build1 !	general overview of configuration information needed to configure AAA to use RADIUS and configure the RADIUS server itself. Note that the FreeRADIUS and AAA passwords must match.
aaa new-model **!**	enables AAA
aaa authentication dot1x default group radius	creates an 802.1X AAA authentication method list

Configuration Item	Description
aaa authorization network default group radius	configures network authorization via RADIUS, including network-related services such as VLAN assignment
aaa accounting identity default start-stop group radius	enables accounting method list for session-aware networking subscriber services
aaa accounting network default start-stop group radius !	enables accounting for all network-related service requests
aaa server radius dynamic-author **client 192.168.11.45 server-key cisco** **server-key cisco** ! aaa session-id common	enables dynamic authorization local server configuration mode and specifies a RADIUS client/key from which a device accepts change of authorization (CoA) and disconnect requests
radius server AAA **address ipv4 192.168.11.45 auth-port 1812**	enables AAA server from the list of multiple AAA servers configured
acct-port 1813 **key cisco**	uses the IP address and ports on which the FreeRADIUS server is listening
ip routing !	
ip dhcp excluded-address 192.168.10.1 192.168.10.100 **!**	DHCP server configuration to exclude selected addresses from pool
ip dhcp pool NCCOE-V3 **network 192.168.13.0 255.255.255.0** **default-router 192.168.13.1** **dns-server 8.8.8.8** **lease 0 12** !	DHCP server configuration to assign IP address to devices on VLAN 3
ip dhcp pool NCCOE-V4 **network 192.168.14.0 255.255.255.0** **default-router 192.168.14.1** **dns-server 8.8.8.8** **!**	DHCP server configuration to assign IP address to devices on VLAN 4
ip dhcp pool NCCOE **network 192.168.10.0 255.255.255.0** **default-router 192.168.10.2** **dns-server 8.8.8.8** **lease 0 12** !	DHCP server configuration to assign IP address to devices on VLAN 1
ip dhcp snooping **ip dhcp snooping vlan 1,3**	enables DHCP snooping globally

Configuration Item	Description
!	specifically enables DHCP snooping on VLANs 1 and 3
access-session attributes filter-list list mudtest **lldp** **dhcp** **access-session accounting attributes filter-spec include list mudtest** **access-session monitor** !	configures access-session attributes to cause LLDP Time Length Values (including the MUD URL) to be forwarded in an accounting message to the AAA server
dot1x logging verbose	global configuration command to filter 802.1x authentication verbose messages
ldp run **!**	enables LLDP, a discovery protocol that runs over layer 2 (the data link layer) to gather information on non-Cisco-manufactured devices
policy-map type control subscriber mud-mab-test **event session-started match-all** **10 class always do-until-failure** **10 authenticate using mab** **!**	configures identity control policies that define the actions that session-aware networking takes in response to specified conditions and subscriber events
template mud-mab-test **switchport mode access** **mab** **access-session port-control auto** **service-policy type control subscriber mud-mab-test** !	enables policy-map (mud-mab-test) and template to cause media access control (MAC) address bypass (MAB) to happen dynamically applies an interface template to a target sets the authorization state of a port. The default value is force-authorized. applies the above previously configured control policy called mud-mab-test
interface GigabitEthernet1/0/13 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/14 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/15 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device

Configuration Item	Description
interface GigabitEthernet1/0/16 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/17 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/18 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/19 **source template mud-mab-test** **!**	statically applies an interface template to a target, i.e., an IoT device
interface GigabitEthernet1/0/20 **source template mud-mab-test**	statically applies an interface template to a target, i.e., an IoT device
interface Vlan1 **ip address 192.168.10.2 255.255.255.0** !	configure and address VLAN1 interface for inter-VLAN routing
interface Vlan2 **ip address 192.168.11.1 255.255.255.0** !	configure and address VLAN2 interface for inter-VLAN routing
interface Vlan3 **ip address 192.168.13.1 255.255.255.0** !	configure and address VLAN3 interface for inter-VLAN routing
interface Vlan4 **ip address 192.168.14.1 255.255.255.0** !	configure and address VLAN4 interface for inter-VLAN routing
interface Vlan5 **ip address 192.168.15.1 255.255.255.0** !	configure and address VLAN5 interface for inter-VLAN routing
! ip default-gateway 192.168.10.1 ip forward-protocol nd ip http server ip http authentication local ip http secure-server ip route 0.0.0.0 0.0.0.0 192.168.10.1 ip route 192.168.12.0 255.255.255.0 192.168.5.1 !	

2.4 DigiCert Certificates

2.4.1 DigiCert CertCentral® Overview

DigiCert's CertCentral® web-based platform allows provisioning and management of publicly trusted X.509 certificates for a variety of purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. For this build, two certificates were provisioned: a private TLS certificate for the MUD file server to support the https connection from the MUD manager to the MUD file server, and a Premium Certificate for signing the MUD files.

2.4.2 Configuration Overview

This section typically documents the network, software, and hardware configurations, but that is not necessary for this component.

2.4.3 Setup

DigiCert allows certificates to be requested through its web-based platform, CertCentral. A user account is needed to access CertCentral. For details on creating a user account and setting up an account, follow the steps described here: https://docs.digicert.com/get-started/

2.4.3.1 TLS Certificate

For this build, we leveraged DigiCert's private TLS certificate because the MUD file server is hosted internally. This certificate supports https connections to the MUD file server, which are required by the MUD manager. Additional information about the TLS certificates offered by DigiCert can be found at https://www.digicert.com/security-certificate-support/.

For instructions on how to order a TLS certificate, proceed to the DigiCert documentation found here, and follow the process for the specific TLS certificate being requested: https://docs.digicert.com/manage-certificates/order-your-ssltls-certificates/

Once requested, integrate the certificate onto the MUD file server as described in Section 2.2.3.1.

2.4.3.2 Premium Certificate

To sign MUD files according to the MUD specification, a client certificate is required. For this implementation, we leveraged DigiCert's Premium Certificate to sign MUD files. This certificate supports signing or encrypting Secure/Multipurpose Internet Mail Extensions messages, which is required by the specification.

For detailed instructions on how to request and implement a Premium Certificate, proceed to the DigiCert documentation found here: https://docs.digicert.com/manage-certificates/client-certificates-guide/.

Once requested, sign MUD files as described in Section 2.2.3.2.2.

2.5 IoT Devices

2.5.1 Molex PoE Gateway and Light Engine

This section provides configuration details of the MUD-capable Molex PoE Gateway and Light Engine used in the build. This component emits a MUD URL that uses LLDP.

2.5.1.1 Configuration Overview

The Molex PoE Gateway runs firmware created and provided by Molex. This firmware was modified by Molex to emit a MUD URL that uses an LLDP message.

2.5.1.1.1 Network Configuration

The Molex PoE Gateway is connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.1.1.2 Software Configuration

For this build, the Molex PoE Gateway is configured with Molex's PoE Gateway firmware, version 1.6.1.8.4.

2.5.1.1.3 Hardware Configuration

The Molex PoE Gateway used in this build is model number 180993-0001, dated March 2017.

2.5.1.2 Setup

The Molex PoE Gateway is controlled via the Constrained Application Protocol (CoAP), and CoAP commands were used to ensure that device functionality was maintained during the MUD process.

2.5.1.2.1 DHCP Client Configuration

The device uses the default DHCP client included in the Molex PoE Gateway firmware.

2.5.2 IoT Development Kits–Linux Based

This section provides configuration details for the Linux-based IoT development kits used in the build, which emit MUD URLs by using DHCP. It also provides information regarding a basic IoT application used to test the MUD process.

2.5.2.1 Configuration Overview

The devkits run various flavors of Linux-based operating systems and are configured to emit a MUD URL during a typical DHCP transaction. They also run a Python script that allows the devkits to receive and process commands by using the MQTT protocol, which can be sent to peripherals connected to the devkits.

2.5.2.1.1 Network Configuration

The devkits are connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.2.1.2 Software Configuration

For this build, the Raspberry Pi is configured on Raspbian 9, the Samsung ARTIK 520 is configured on Fedora 24, and the Intel UP Squared Grove is configured on Ubuntu 16.04 LTS. The devkits also utilized dhclient as the default DHCP client. This DHCP client is installed natively on many Linux distributions and can be installed using a preferred package manager if not currently present.

2.5.2.1.3 Hardware Configuration

The hardware used for these devkits included the Raspberry Pi 3 Model B, Samsung ARTIK 520, and Intel UP Squared Grove.

2.5.2.2 Setup

The following subsection describes setting up the devkits to send a MUD URL during the DHCP transaction and to act as a connected device by leveraging an MQTT broker server (we describe setting up the MQTT broker server in Section 2.8).

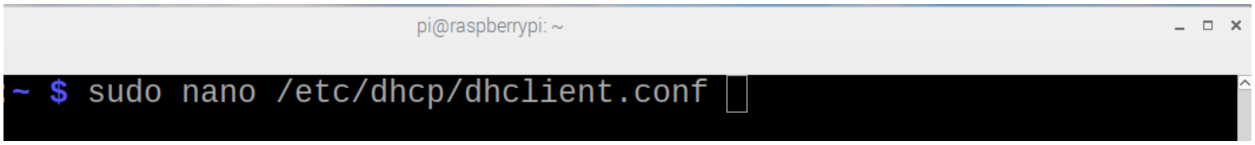
2.5.2.2.1 DHCP Client Configuration

We leveraged dhclient as the default DHCP client for these devices due to the availability of the DHCP client on different Linux platforms and the ease of emitting MUD URLs via DHCP.

To set up the dhclient configuration:

1. Open a terminal on the device.
2. Ensure that any other conflicting DHCP clients are disabled or removed.
3. Install the dhclient package (if needed).
4. Edit the *dhclient.conf* file by entering the following command:

   ```
   sudo nano /etc/dhcp/dhclient.conf
   ```

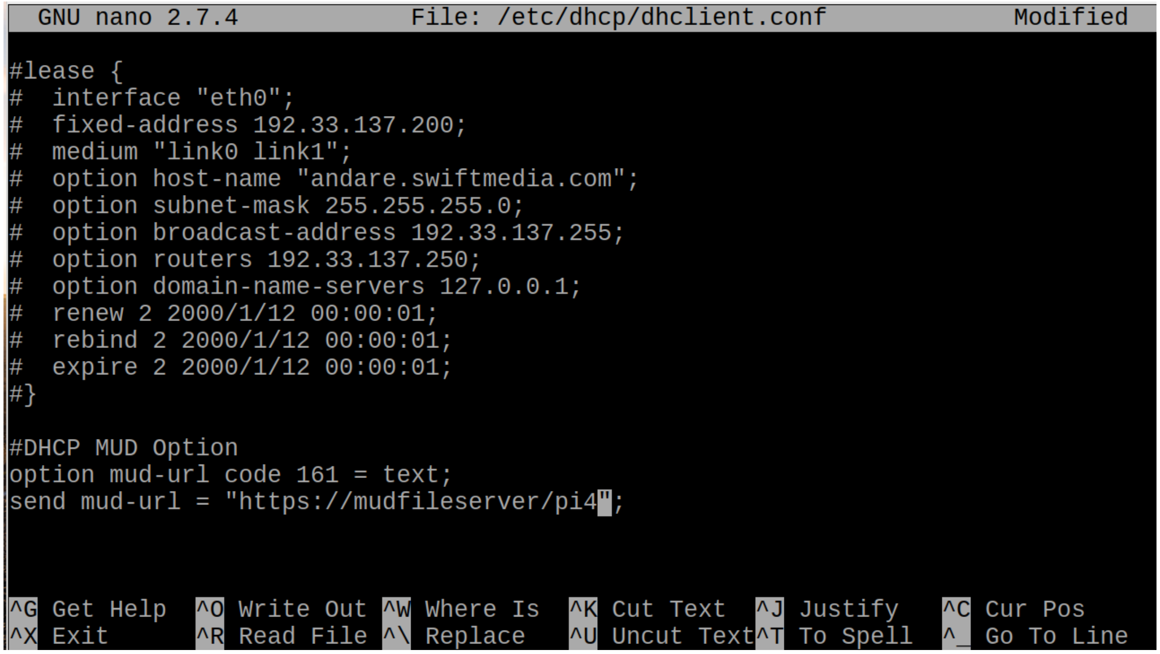
   

   ```
   ~ $ sudo nano /etc/dhcp/dhclient.conf
   ```

5. Add the following lines:

   ```
   option mud-url code 161 = text;
   ```

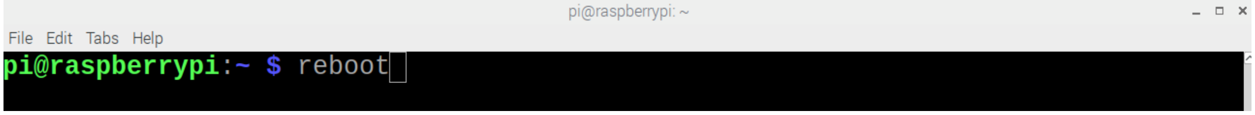
```
send mud-url = "<insert URL for MUD File here>";
```



```
  GNU nano 2.7.4              File: /etc/dhcp/dhclient.conf              Modified

#lease {
#  interface "eth0";
#  fixed-address 192.33.137.200;
#  medium "link0 link1";
#  option host-name "andare.swiftmedia.com";
#  option subnet-mask 255.255.255.0;
#  option broadcast-address 192.33.137.255;
#  option routers 192.33.137.250;
#  option domain-name-servers 127.0.0.1;
#  renew 2 2000/1/12 00:00:01;
#  rebind 2 2000/1/12 00:00:01;
#  expire 2 2000/1/12 00:00:01;
#}

#DHCP MUD Option
option mud-url code 161 = text;
send mud-url = "https://mudfileserver/pi4";
```

6. Save and close the file.
7. Reboot the device:

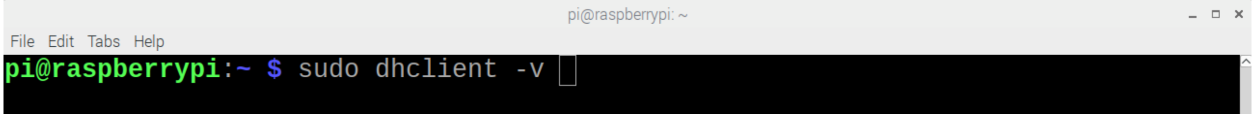
```
Reboot
```



```
pi@raspberrypi:~ $ reboot
```

8. Open a terminal.
9. Execute the dhclient:

```
sudo dhclient -v
```



```
pi@raspberrypi:~ $ sudo dhclient -v
```

2.5.2.2.2 IoT Application for Testing

The following Python application was created by the NCCoE to enable the devkits to act as basic IoT devices:

```
#Program:                 IoTapp.
```

```
#Version:                  1.0
#Purpose:                  Provide IoT capabilities to devkit.
#Protocols:         MQTT.
#Functionality:     Allow remote control of LEDs on connected breadboard.

#Libraries
import paho.mqtt.client as mqttClient
import time
import RPi.GPIO as GPIO

#Global Variables
BrokerAddress = "192.168.1.87"   #IP address of Broker(Server), change as needed. Best
practice would be a registered domain name that can be queried for appropriate server
address.
BrokerPort = "1883"        #Default port used by most MQTT Brokers. Would be 1883 if
using Transport Encryption with TLS.
ConnectionStatus = "Disconnected" #Status of connection to Broker. Should be either
"Connected" or "Disconnected".
LED = 26

#Supporting Functions
def on_connect(client, userdata, flags, rc):   #Function for connection status to
Broker.
    if rc == 0:
        ConnectionStatus = "Connected to Broker!"
        print(ConnectionStatus)
    else:
        ConnectionStatus = "Connection Failed!"
        print(ConnectionStatus)

def on_message(client, userdata, msg):         #Function for parsing message data.
    if "ON" in msg.payload:
        print("ON!")
        GPIO.output(LED, 1)

    if "OFF" in msg.payload:
        print("OFF!")
        GPIO.output(LED, 0)

def  MQTTapp():
    client = mqttClient.Client()     #New instance.
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(BrokerAddress, BrokerPort)
    client.loop_start()
    client.subscribe("test")
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("8")
        client.disconnect()
```

```
                client.loop_stop()

#Main Function
def main():

        GPIO.setmode(GPIO.BCM)
        GPIO.setup(LED, GPIO.OUT)

        print("Main function has been executed!")
        MQTTapp()

if __name__ == "__main__":
        main()
```

2.5.3 IoT Development Kit–u-blox C027-G35

This section details configuration of a u-blox C027-G35, which emits a MUD URL by using DHCP, and a basic IoT application used to test MUD rules.

2.5.3.1 Configuration Overview

This devkit runs the Arm Mbed-OS and is configured to emit a MUD URL during a typical DHCP transaction. It also runs a basic IoT application to test MUD rules.

2.5.3.1.1 Network Configuration

The u-blox C027-G35 is connected to the network over a wired Ethernet connection. The IP address is assigned dynamically by using DHCP.

2.5.3.1.2 Software Configuration

For this build, the u-blox C027-G35 was configured on the Mbed-OS 5.10.4 operating system.

2.5.3.1.3 Hardware Configuration

The hardware used for this devkit is the u-blox C027-G35.

2.5.3.2 Setup

The following subsection describes setting up the u-blox C027-G35 to send a MUD URL in the DHCP transaction and to act as a connected device by establishing network connections to the update server and other destinations.

2.5.3.2.1 DHCP Client Configuration

To add MUD functionality to the Mbed-OS DHCP client, the following two files inside Mbed-OS require modification:

- *mbed-os/features/lwipstack/lwip/src/include/lwip/prot/dhcp.h*
    - **NOT** *mbed-os/features/lwipstack/lwip/src/include/lwip/dhcp.h*

- *mbed-os/features/lwipstack/lwip/src/core/ipv4/lwip_dhcp.c*

Changes to include/lwip/prot/dhcp.h:

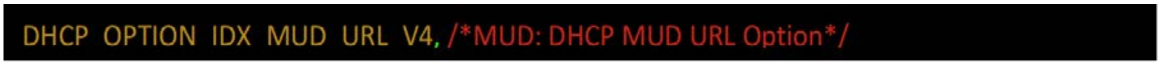
1. Add the following line below the greatest DCHP option number (67) on line 170:

```
#define DHCP_OPTION_MUD_URL_V4    161 /*MUD: RFC-ietf-opsawg-mud-25 draft-ietf-opsawg-mud-08,
Manufacturer Usage Description*/
```

Changes to core/ipv4/lwip_dhcp.c:

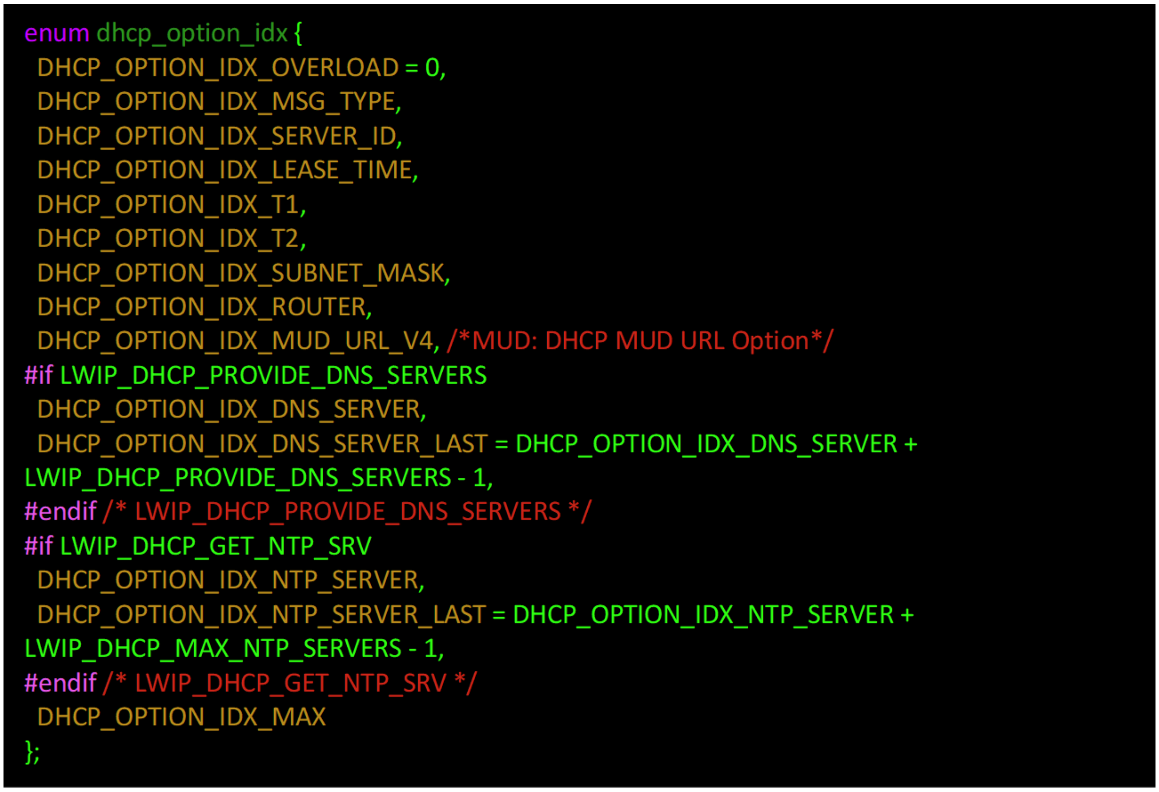
1. Change within container around line 141:

   To enum dhcp_option_idx (at line 141) before the first #if, add



```
DHCP  OPTION  IDX  MUD  URL  V4, /*MUD: DHCP MUD URL Option*/
```

   It should now look like the screenshot below:



```
enum dhcp_option_idx {
  DHCP_OPTION_IDX_OVERLOAD = 0,
  DHCP_OPTION_IDX_MSG_TYPE,
  DHCP_OPTION_IDX_SERVER_ID,
  DHCP_OPTION_IDX_LEASE_TIME,
  DHCP_OPTION_IDX_T1,
  DHCP_OPTION_IDX_T2,
  DHCP_OPTION_IDX_SUBNET_MASK,
  DHCP_OPTION_IDX_ROUTER,
  DHCP_OPTION_IDX_MUD_URL_V4, /*MUD: DHCP MUD URL Option*/
#if LWIP_DHCP_PROVIDE_DNS_SERVERS
  DHCP_OPTION_IDX_DNS_SERVER,
  DHCP_OPTION_IDX_DNS_SERVER_LAST = DHCP_OPTION_IDX_DNS_SERVER +
LWIP_DHCP_PROVIDE_DNS_SERVERS - 1,
#endif /* LWIP_DHCP_PROVIDE_DNS_SERVERS */
#if LWIP_DHCP_GET_NTP_SRV
  DHCP_OPTION_IDX_NTP_SERVER,
  DHCP_OPTION_IDX_NTP_SERVER_LAST = DHCP_OPTION_IDX_NTP_SERVER +
LWIP_DHCP_MAX_NTP_SERVERS - 1,
#endif /* LWIP_DHCP_GET_NTP_SRV */
  DHCP_OPTION_IDX_MAX
};
```

2. Change within the function around line 975:

a. To the list of local variables for `static err_t dhcp_discover(struct netif *netif)`, add the desired MUD URL (*www.example.com* used here):

```
char* mud_url = "https://www.example.com";    /*MUD: MUD URL*/
```

NOTE: The MUD URL must be less than 255 octets/bytes/characters long.

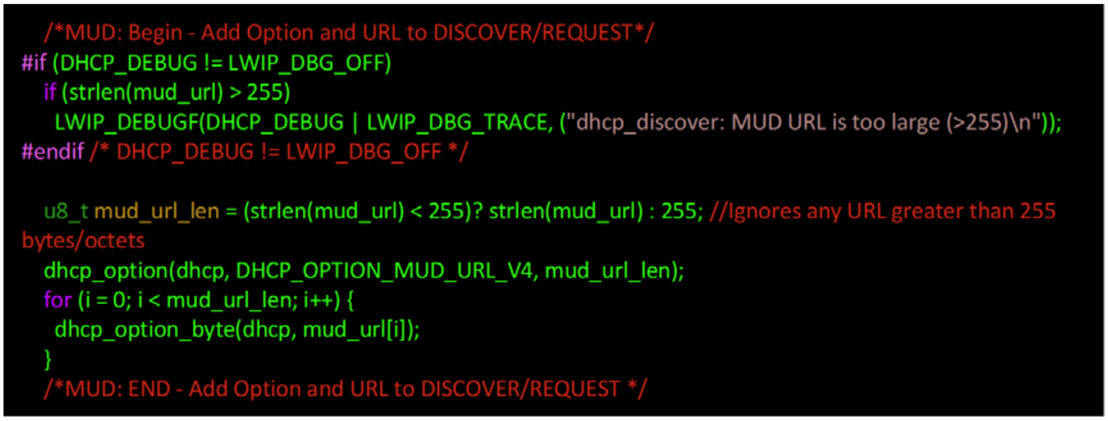
b. Within if (result == ERR_OK) after

```
dhcp_option(dhcp, DHCP_OPTION_PARAMETER_REQUEST_LIST,
LWIP_ARRAYSIZE(dhcp_discover_request_options));
    for (i = 0; i < LWIP_ARRAYSIZE(dhcp_discover_request_options); i++) {
      dhcp_option_byte(dhcp, dhcp_discover_request_options[i]);
    }
```

and before:

```
    dhcp_option_trailer(dhcp);
```

add:



```
   /*MUD: Begin - Add Option and URL to DISCOVER/REQUEST*/
#if (DHCP_DEBUG != LWIP_DBG_OFF)
   if (strlen(mud_url) > 255)
     LWIP_DEBUGF(DHCP_DEBUG | LWIP_DBG_TRACE, ("dhcp_discover: MUD URL is too large (>255)\n"));
#endif /* DHCP_DEBUG != LWIP_DBG_OFF */

   u8_t mud_url_len = (strlen(mud_url) < 255)? strlen(mud_url) : 255; //Ignores any URL greater than 255
bytes/octets
   dhcp_option(dhcp, DHCP_OPTION_MUD_URL_V4, mud_url_len);
   for (i = 0; i < mud_url_len; i++) {
     dhcp_option_byte(dhcp, mud_url[i]);
   }
   /*MUD: END - Add Option and URL to DISCOVER/REQUEST */
```

3. Change within the function around line 1486:

Within the following function:

```
static err_t
dhcp_parse_reply(struct dhcp *dhcp, struct pbuf *p)
```

Within switch(op) before default, add the following case (around line 1606):

```
case(DHCP_OPTION_MUD_URL_V4): /* MUD Testing */
 LWIP_ERROR("len == 0", len == 0, return ERR_VAL;);
 decode_idx = DHCP_OPTION_IDX_MUD_URL_V4;
 break;
```

4. Compile by using the following command:

```
mbed compile -m ublox_c027 -t gcc_arm
```

2.5.3.2.2 IoT Application for Testing

The following application was created by the NCCoE to enable the devkit to test the build as a MUD-capable device:

```
#include "mbed.h"
#include "EthernetInterface.h"

//DigitalOut led1(LED1);
PwmOut led2(LED2);
Serial pc(USBTX, USBRX);

float brightness = 0.0;

// Network interface
EthernetInterface net;

// Socket demo
int main() {
  int led1 = true;

  for (int i = 0; i < 4; i++) {

    led2 = (led1)? 0.5 : 0.0;

    led1 = !led1;
    wait(0.5);
  }

  for (int i = 0; i < 8; i++) {

    led2 = (led1)? 0.5 : 0.0;

    led1 = !led1;
    wait(0.25);
  }

  for (int i = 0; i < 8; i++) {

    led2 = (led1)? 0.5 : 0.0;

    led1 = !led1;
    wait(0.125);
  }
  TCPSocket socket;
  char sbuffer[] = "GET / HTTP/1.1\r\nHost: www.updateserver.com\r\n\r\n";
  char bbuffer[] = "GET / HTTP/1.1\r\nHost: www.unapprovedserver.com\r\n\r\n";
  int scount, bcount;
  char rbuffer[64];
  char brbuffer[64];
  int rcount, brcount;

  /* By default grab an IP address*/
  // Bring up the ethernet interface
  pc.printf("Ethernet socket example\r\n");
  net.connect();
  // Show the network address
  const char *ip = net.get_ip_address();
  pc.printf("IP address is: %s\r\n", ip ? ip : "No IP");
  socket.open(&net);
  /* End of default IP address */

  pc.printf("Press U to turn LED1 brightness up, D to turn it down, G to get IP, R to
release IP, H for HTTP request, B for blocked HTTP request\r\n");

  while(1) {
    char c = pc.getc();
    if((c == 'u') && (brightness < 0.5)) {
      brightness += 0.01;
      led2 = brightness;
    }
    if((c == 'd') && (brightness > 0.0)) {
      brightness -= 0.01;
      led2 = brightness;
    }
    if(c == 'g'){
      // Bring up the ethernet interface
      pc.printf("Sending DHCP Request...\r\n");
      net.connect();
      // Show the network address
      const char *ip = net.get_ip_address();
      pc.printf("IP address is: %s\r\n", ip ? ip : "No IP");
    }
    if(c == 'r'){
      socket.close();
      net.disconnect();
      pc.printf("IP Address Released\r\n");
    }
    if(c == 'h'){

     pc.printf("Sending HTTP Request...\r\n");
     // Open a socket on the network interface, and create a TCP connection
     socket.open(&net);
     socket.connect("www.updateserver.com", 80);
     // Send a simple http request
     scount = socket.send(sbuffer, sizeof sbuffer);
     pc.printf("sent %d [%.*s]\r\n", scount, strstr(sbuffer, "\r\n")-sbuffer, sbuffer);
     // Receive a simple http response and print out the response line
     rcount = socket.recv(rbuffer, sizeof rbuffer);
     pc.printf("recv %d [%.*s]\r\n", rcount, strstr(rbuffer, "\r\n")-rbuffer, rbuffer);
     socket.close();
    }
    if(c == 'b'){
     pc.printf("Sending Blocked HTTP Request...\r\n");
     // Open a socket on the network interface, and create a TCP connection
     socket.open(&net);
     socket.connect("www.unapprovedserver.com", 80);
     // Send a simple http request
     bcount = socket.send(bbuffer, sizeof bbuffer);
     pc.printf("sent %d [%.*s]\r\n", bcount, strstr(bbuffer, "\r\n")-bbuffer, bbuffer);

     // Receive a simple http response and print out the response line
     brcount = socket.recv(brbuffer, sizeof brbuffer);
     pc.printf("recv %d [%.*s]\r\n", brcount, strstr(brbuffer, "\r\n")-brbuffer,
brbuffer);
     socket.close();
    }
  }
}
```

2.5.4 IoT Devices–Non-MUD-Capable

This section details configuration of non-MUD-capable IoT devices attached to the implementation network. These include several types of devices, such as cameras, mobile phones, lighting, a connected assistant, a printer, a baby monitor, a wireless access point, and a digital video recorder. These devices did not emit a MUD URL or have MUD capabilities of any kind.

2.5.4.1 Configuration Overview

These non-MUD-capable IoT devices are unmodified and still retain the default manufacturer configurations.

2.5.4.1.1 Network Configuration

These IoT devices are configured to obtain an IP address via DHCP.

2.5.4.1.2 Software Configuration

The software on these devices is configured according to standard manufacturer instructions.

2.5.4.1.3 Hardware Configuration

The hardware used in these devices is unmodified from manufacturer specifications.

2.5.4.2 Setup

These devices were set up according to the manufacturer instructions and connected to the Cisco switch via Ethernet cable or connected wirelessly through the wireless access point.

2.5.4.2.1 DHCP Client Configuration

These IoT devices used the default DHCP clients provided by the original manufacturer and were not modified in any way.

2.6 Update Server

This section describes how to implement a server that will act as an update server. It will attempt to access and be accessed by the IoT device, in this case one of the development kits we built in the lab.

2.6.1 Update Server Overview

The update server is an Apache web server that hosts mock software update files to be served as software updates to our IoT device devkits. When the server receives an http request, it sends the corresponding update file.

2.6.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the update server.

2.6.2.1 Network Configuration

The IP address was statically assigned.

2.6.2.2 Software Configuration

For this build, the update server was configured on the Ubuntu 18.04 LTS operating system.

2.6.2.3 Hardware Configuration

The update server was hosted in the NCCoE's virtual environment, functioning as a cloud service.

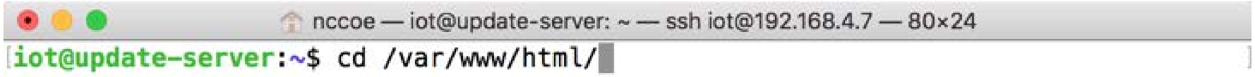
2.6.3 Setup

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. After completing the process, the SSL/TLS encryption was set up by using the digital certificate and key obtained from DigiCert. This was set up by using the official Apache documentation, found at https://httpd.apache.org/docs/current/ssl/ssl_howto.html.

The following configurations were made to the server to host the update file:

1. Open a terminal.
2. Change directories to the Hypertext Markup Language (HTML) folder:

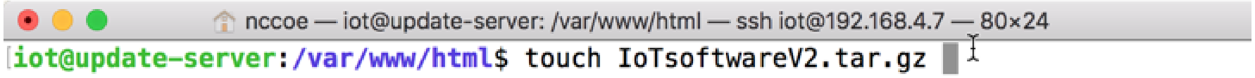
```
cd /var/www/html/
```



```
iot@update-server:~$ cd /var/www/html/
```

3. Create the update file (Note: this is a mock update file):

```
touch IoTsoftwareV2.tar.gz
```


nccoe — iot@update-server: /var/www/html — ssh iot@192.168.4.7 — 80×24

```
iot@update-server:/var/www/html$ touch IoTsoftwareV2.tar.gz
```

2.7 Unapproved Server

This section describes how to implement a server that will act as an unapproved server. It will attempt to access and to be accessed by an IoT device, in this case one of the MUD-capable devices on the implementation network.

2.7.1 Unapproved Server Overview

The unapproved server is an internet host that is not explicitly authorized in the MUD file to communicate with the IoT device. When the IoT device attempts to connect to this server, the router or switch should not allow this traffic because it is not an approved internet service as defined by the corresponding MUD file. Likewise, when the server attempts to connect to the IoT device, this traffic should be denied at the router or switch.

2.7.2 Configuration Overview

The following subsections document the software, hardware, and network configurations for the unapproved server.

2.7.2.1 Network Configuration

The unapproved server hosts a web server that is accessed via transmission control protocol (TCP) port 80. Any applications that request access to this server need to be able to connect on this port. Use firewall-cmd, iptables, or any other system utility for manipulating the firewall to open this port.

2.7.2.2 Software Configuration

For this build, the CentOS 7 OS was leveraged with an Apache web server.

2.7.2.3 Hardware Configuration

The unapproved server was hosted in the NCCoE's virtual environment, functioning as a cloud service. The IP address was statically assigned.

2.7.3 Setup

The following subsection describes the setup process for configuring the unapproved server.

2.7.3.1 Apache Web Server

The Apache web server was set up by using the official Apache documentation at https://httpd.apache.org/docs/current/install.html. SSL/TLS encryption was not used for this server.

2.8 MQTT Broker Server

2.8.1 MQTT Broker Server Overview

For this build, the open-source tool Mosquitto was used as the MQTT broker server. The server communicates publish and subscribe messages among multiple clients. For our implementation, this server allows mobile devices set up with the appropriate application to communicate with the MQTT-enabled IoT devices in the build. The messages exchanged by the devices are on and off messages, which allow the mobile device to control the LED light on the MQTT-enabled IoT device.

2.8.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the MQTT broker server.

2.8.2.1 Network Configuration

The MQTT broker server was hosted in the NCCoE’s virtual environment, functioning as a cloud service. The IP address was statically assigned.

The server is accessed via TCP port 1883. Any clients that require access to this server need to be able to connect on this port. Use firewall-cmd, iptables, or any other system utility for manipulating the firewall to open this port.

2.8.2.2 Software Configuration

For this build, the MQTT broker server was configured on an Ubuntu 18.04 LTS operating system.

2.8.2.3 Hardware Configuration

This server was hosted in the NCCoE’s virtual environment, functioning as a cloud service. The IP address was statically assigned.

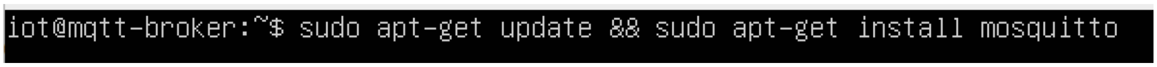
2.8.3 Setup

In this section we describe setting up the MQTT broker server to communicate messages to and from the controlling application and the IoT device.

2.8.3.1 Mosquitto Setup

1. Install the open-source MQTT broker server, Mosquitto, by entering the following command:

   ```
   sudo apt-get update && sudo apt-get install mosquitto
   ```

   ```
   iot@mqtt-broker:~$ sudo apt-get update && sudo apt-get install mosquitto
   ```

Following the installation, this implementation leveraged the default configuration of the Mosquitto server. The MQTT broker server was set up by using the official Mosquitto documentation at https://mosquitto.org/man/.

2.9 Forescout–IoT Device Discovery

This section describes how to implement Forescout's appliance and enterprise manager to provide device discovery on the network.

2.9.1 Forescout Overview

The Forescout appliance discovers, catalogs, profiles, and classifies the devices that are connected to the demonstration network. When a device is added to or removed from the network, the Forescout appliance is updated and actively monitors these devices on the network. The administrator will be able to manage multiple Forescout appliances from a central point by integrating the appliance with the enterprise manager.

2.9.2 Configuration Overview

The following subsections document the software, hardware, and network requirements for the Forescout appliance and enterprise manager.

2.9.2.1 Network Configuration

The virtual Forescout appliance was hosted on VLAN 2 of the Cisco switch. It was set up with just the monitor interface. The network configuration for the Forescout appliance was completed by using the official Forescout documentation at https://www.Forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 2 and 8).

The virtual enterprise manager was hosted in the virtual environment that is shared across each build.

2.9.2.2 Software Configuration

The build leveraged a virtual Forescout appliance VCT-R version 8.0.1 along with a virtual enterprise manager VCEM-05 version 8.0.1. Both virtual appliances were built on a Linux OS supported by Forescout.

Forescout provides software for managing the appliances on the network. The Forescout console is software that allows management of the Forescout appliance/enterprise manager and visualization of the data gathered by the appliances.

2.9.2.3 Hardware Configuration

The build leveraged a virtual Forescout appliance, which was set up in the lab environment on a dedicated machine hosting the local virtual machines in Build 1.

The virtual enterprise manager was hosted in the NCCoE's virtual environment with a static IP assignment.

2.9.3 Setup

In this section we describe setting up the virtual Forescout appliance and the virtual enterprise manager.

2.9.3.1 Forescout Appliance Setup

The virtual Forescout appliance was set up by using the official Forescout documentation at https://www.Forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 3 and 8).

2.9.3.2 Enterprise Manager Setup

The enterprise manager was set up by using the official Forescout documentation at https://www.Forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf (see Chapters 4 and 8).

Using the enterprise manager, we configured the following modules:

- Endpoint
- Network
- Authentication
- Core Extension
- Device Profile Library—https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Device_Profile_Library.pdf

- IoT Posture Assessment Library—https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_IoT_Posture_Assessment_Library-1.pdf
- Network Interface Card (NIC) Vendor DB—https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_NIC_Vendor_DB_17.0.12.pdf
- Windows Applications—https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Applications.pdf
- Windows Vulnerability Database (DB)—https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Vulnerability_DB_18.0.2.pdf
- Open Integration Module—https://www.Forescout.com/wp-content/uploads/2018/08/CounterACT_Open_Integration_Module_Overview_1.1.pdf

3 Build 2 Product Installation Guides

This section of the practice guide contains detailed instructions for installing and configuring the products used to implement Build 2. For additional details on Build 2's logical and physical architectures, please refer to NIST SP 1800-15B.

3.1 Yikes! MUD Manager

This section describes the Yikes! MUD manager version v1.1.3, which is a software package deployed on the Yikes! router. It should not require configuration as it should be fully functioning upon connecting the Yikes! router to the network.

3.1.1 Yikes! MUD Manager Overview

The Yikes! MUD manager is a software package supported by MasterPeace within the Yikes! physical router. The version of the Yikes! router used in this implementation supports IoT devices that leverage DHCP as their default MUD emission method.

3.1.2 Configuration Overview

At this implementation, no additional network, software, or hardware configuration was required to enable the Yikes! MUD manager capability on the Yikes! router.

3.1.3 Setup

At this implementation, no setup was required to enable the Yikes! MUD manager capability on the Yikes! router. See the Yikes! Router section for details on the router setup.

3.2 MUD File Server

3.2.1 MUD File Server Overview

For this build, the NCCoE leveraged a MUD file server hosted by MasterPeace. This file server hosts MUD files along with their corresponding signature files for the MUD-capable IoT devices used in Build 2. The MUD file server is responsible for serving the MUD file and the corresponding signature file upon request from the MUD manager. These files were created by the NCCoE and provided to MasterPeace to host due to the Yikes! cloud component requirement that the MUD file server be internet accessible to display the contents of the MUD file in the Yikes! user interface (UI).

To build an on-premises MUD file server and to create MUD files for MUD-capable IoT devices, please follow the instructions in Build 1's MUD File Server section.

3.3 Yikes! DHCP Server

This section describes the Yikes! DHCP server, which should also be fully functional out of the box and should not require any modification upon receipt.

3.3.1 Yikes! DHCP Server Overview

The Yikes! DHCP server is MUD capable and, like the Yikes! MUD manager and Yikes! threat-signaling agent, is a logical component within the Yikes! router. In addition to dynamically assigning IP addresses, it recognizes the DHCP option (161) and logs DHCP events that include this option to a log file. This log file is monitored by the Yikes! MUD manager, which is responsible for handling the MUD requests.

3.3.2 Configuration Overview

At this implementation, no additional network, software, or hardware configuration was required to enable the Yikes! DHCP server capability on the Yikes! router.

3.3.3 Setup

At this implementation, no additional setup was required.

3.4 Yikes! Router

This section describes how to implement and configure the Yikes! router, which requires minimal configuration from a user standpoint.

3.4.1 Yikes! Router Overview

The Yikes! router is a customized original equipment manufacturer product, which at implementation was a preproduction product. It is a self-contained router, Wi-Fi access point, and firewall that communicates locally with Wi-Fi devices and wired devices. The Yikes! router leveraged in this implementation was developed on an OpenWRT base router with the Yikes! capabilities added on. The Yikes! router hosts all the software necessary to enable a MUD infrastructure on premise. It also communicates with the Yikes! cloud and threat-signaling services to support additional capabilities in the network.

At this implementation, the Yikes! MUD manager, DHCP server, and GCA threat-signaling components all reside on the Yikes! router and are configured to function without any additional configuration.

3.4.2 Configuration Overview

3.4.2.1 Network Configuration

Implementation of a Yikes! router requires an internet source such as a Digital Subscriber Line (DSL) or cable modem.

3.4.2.2 Software Configuration

At this implementation, no additional software configuration was required to set up the Yikes! router.

3.4.2.3 Hardware Configuration

At this implementation, no additional hardware configuration was required to set up the Yikes! router.

3.4.3 Setup

As stated earlier, the version of the Yikes! router used in Build 2 was preproduction, so MasterPeace may have performed some setup and configuration steps that are not documented here. Those additional steps, however, are not expected to be required to set up the production version of the router. The following setup steps were performed:

1. Unbox the Yikes! router and provided accessories.
2. Connect the Yikes! router’s wide area network port to an internet source (e.g., cable modem or DSL).
3. Plug the power supply into the Yikes! router.
4. Power on the Yikes! router.

After powering on the router, the network password must be provided so the router can authenticate itself to the network. In addition, best security practices (not documented here), such as changing the router's administrative password, should be followed in accordance with the security policies of the user.

3.5 DigiCert Certificates

DigiCert's CertCentral web-based platform allows provisioning and management of publicly trusted X.509 certificates for a variety of purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. For Build 2, the Premium Certificate created in Build 1 was leveraged for signing the MUD files. To request and implement DigiCert certificates, follow the documentation in Build 1's DigiCert Certificates section and subsequent sections.

3.6 IoT Devices

3.6.1 IoT Development Kits—Linux Based

3.6.1.1 Configuration Overview

This section provides configuration details for the Linux-based IoT development kits used in the build, which emit MUD URLs by using DHCP. It also provides information regarding a basic IoT application used to test the MUD process.

3.6.1.1.1 Network Configuration

The devkits are connected to the network over both a wired Ethernet connection and wirelessly. The IP address is assigned dynamically by using DHCP.

3.6.1.1.2 Software Configuration

For this build, Raspberry Pi is configured on Raspbian 9, the Samsung ARTIK 520 is configured on Fedora 24, the NXP i.MX 8m is configured on Yocto Linux, and the BeagleBone Black is configured on Debian 9.5. The devkits also utilized a variety of DHCP clients, including dhcpcd and dhclient (see Build 1's IoT Development Kits–Linux Based section for dhclient configurations). This build introduced dhcpcd as a method for emitting a MUD URL for all devkits in this build, apart from the NXP i.MX 8m, which leveraged dhclient. Dhcpcd is installed natively on many Linux distributions and can be installed using a preferred package manager if not currently present.

3.6.1.1.3 Hardware Configuration

The hardware used for these devkits included the Raspberry Pi 3 Model B, Samsung ARTIK 520, NXP i.MX 8m, and BeagleBone Black.

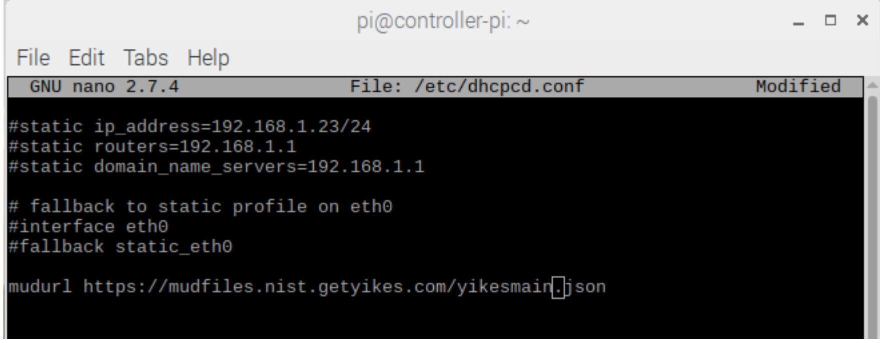
3.6.1.2 Setup

The following subsection describes setting up the devkits to send a MUD URL during the DHCP transaction using dhcpcd as the DHCP client on the Raspberry Pi. For dhclient instructions, see Build 1's Setup and DHCP Client Configuration sections.

3.6.1.2.1 DHCP Client Configuration

These devkits utilized dhcpcd version 7.2.3. Configuration consisted of adding the following line to the file located at */etc/dhcpcd.conf*:

```
mudurl https://<example-url>
```



```
  GNU nano 2.7.4                File: /etc/dhcpcd.conf                Modified

#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

mudurl https://mudfiles.nist.getyikes.com/yikesmain.json
```

3.7 Update Server

Build 2 leveraged the preexisting update server that is described in Build 1's Update Server section. To implement a server that will act as an update server, see the documentation in Build 1's Update Server section. The update server will attempt to access and be accessed by the IoT device, which, in this case, is one of the development kits we built in the lab.

3.8 Unapproved Server

Build 2 leverages the preexisting unapproved server that is described in Build 1's Unapproved Server section. To implement a server that will act as an unapproved server, see the documentation in Build 1's Unapproved Server section. The unapproved server will attempt to access and to be accessed by an IoT device, which, in this case, is one of the MUD-capable devices on the implementation network.

3.9 Yikes! IoT Device Discovery, Categorization, and Traffic Policy Enforcement (Yikes! Cloud and Yikes! Mobile Application)

This section describes how to implement and configure Yikes! IoT device discovery, categorization, and traffic policy enforcement, which is a capability supported by the Yikes! router, Yikes! cloud, and Yikes! mobile application.

3.9.1 Yikes! IoT Device Discovery, Categorization, and Traffic Policy Enforcement Overview

The Yikes! router provides an IoT device discovery service for Build 2. Yikes! discovers, inventories, profiles, and classifies devices connected to the local network consistent with each device's type and allows traffic enforcement policies to be configured by the user through the Yikes! mobile application.

Yikes! isolates every device on the network so that, by default, no device is permitted to communicate with any other device. Devices added to the network are automatically identified and categorized based on information such as DHCP header, MAC address, operating system, manufacturer, and model.

Using the Yikes! mobile application, users can define fine-grained device filtering. The enforcement can be set to enable specific internet access (north/south) and internal network access to specific devices (east/west) as determined by category-specific rules.

3.9.2 Configuration Overview

3.9.2.1 Network Configuration

No network configurations outside Yikes! router network configurations are required to enable this capability.

3.9.2.2 Software Configuration

MasterPeace performed some software configuration on the Yikes! router after it was deployed as part of Build 2. Aside from this, no additional software configuration was required to support device discovery. When the production version of the Yikes! router is available, it is not expected to require configuration. The Yikes! mobile application was still in development during deployment. The build used the web-based Yikes! mobile application from a laptop in the lab environment to display and configure device information and traffic policies.

3.9.2.3 Hardware Configuration

At this implementation, the Yikes! mobile application was not published in an application store. For this reason, a desktop was leveraged to load the web page hosting the "mobile application."

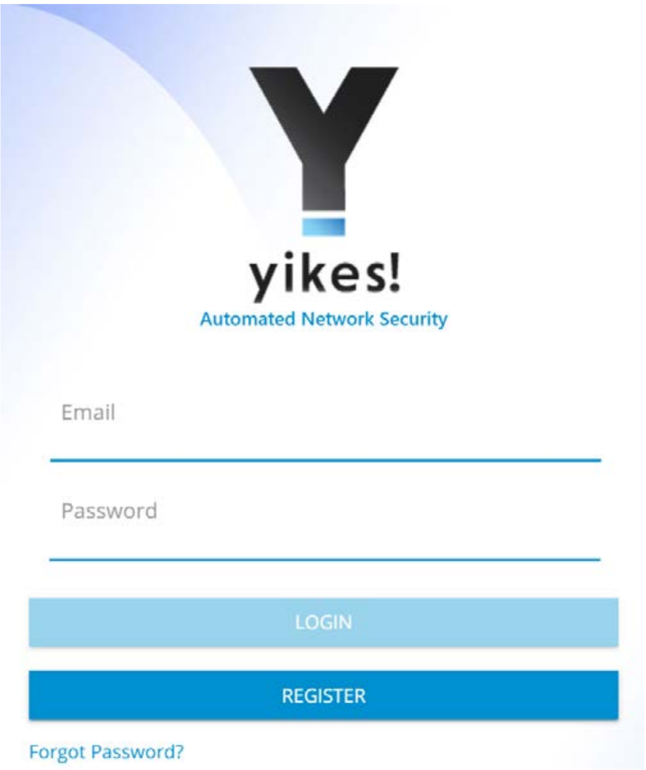
3.9.3 Setup

Once devices have been added to the network on the Yikes! router, they will appear in the Yikes! cloud inventory, which is accessible via the Yikes! mobile application. At this implementation, the Yikes! mobile application and the processes associated with the Yikes! cloud service were under development. It is possible that the design of the UI and the workflow will change for the final implementation of the mobile application.

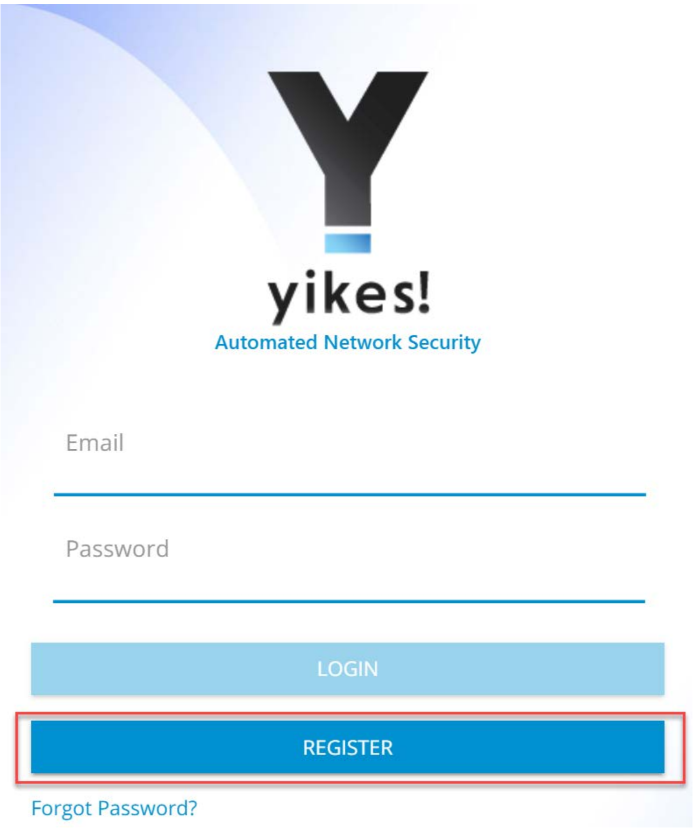
3.9.3.1 Yikes! Router and Account Cloud Registration

At this implementation, the Yikes! router and cloud account registration processes were under development. As a result, this section will not describe how to associate a Yikes! router with a Yikes! cloud instance. The steps below show the process for account registration at this implementation.

1. Open a browser and access the Yikes! UI (In the preproduction version of the router, accessing the UI required inputting a URL provided by MasterPeace):

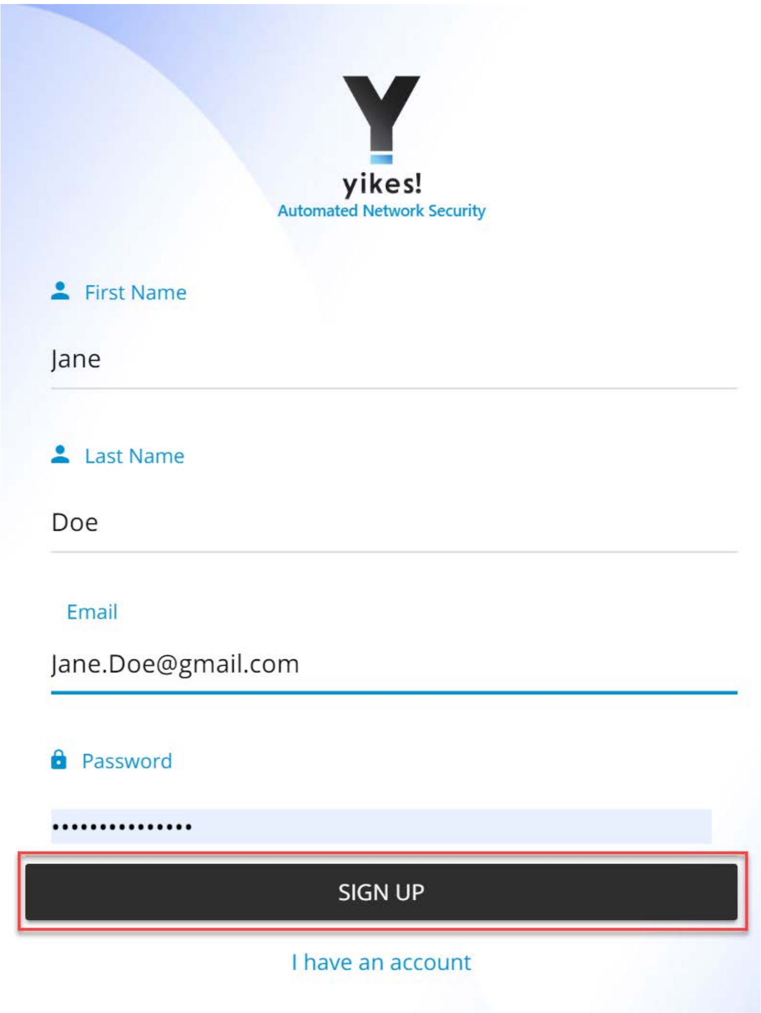


1532 2. Click on the **Register** button to sign up for an account:



1533

3. Populate the requested information for the account: First Name, Last Name, Email, and Password. Click **Sign Up:**



First Name
Jane

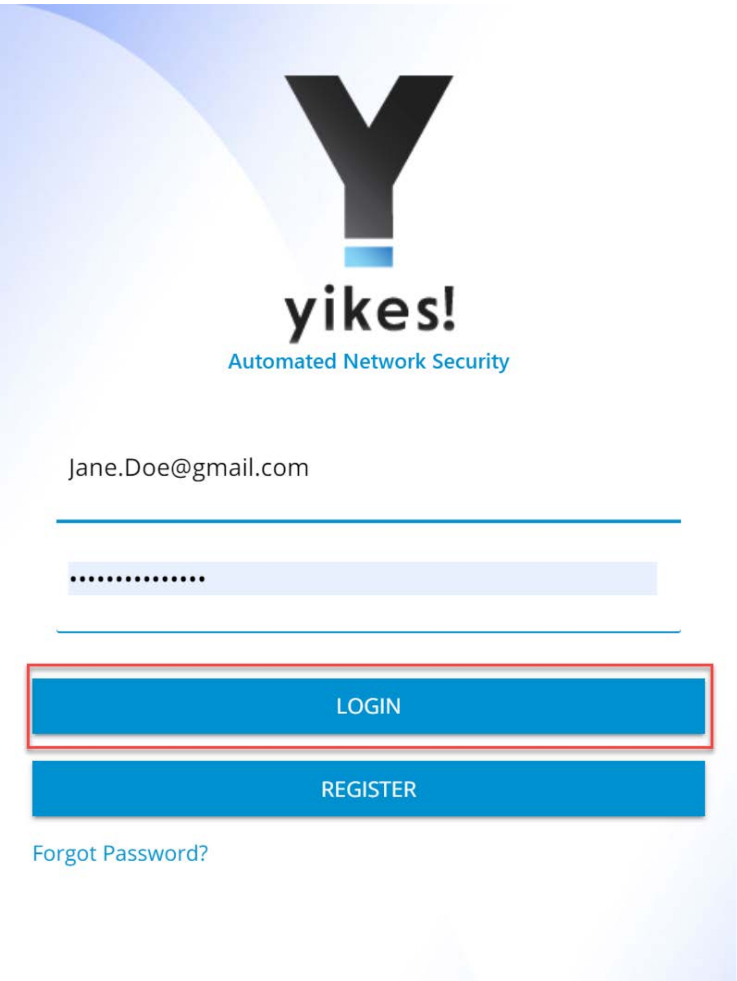
Last Name
Doe

Email
Jane.Doe@gmail.com

Password
•••••••••••••••

Note: There will be additional steps related to associating the Yikes! router with the Yikes! account being created. However, at this implementation, this process was still under development.

4. Once the account is approved and linked to the Yikes! router, **Log in** with the credentials created in step 3:



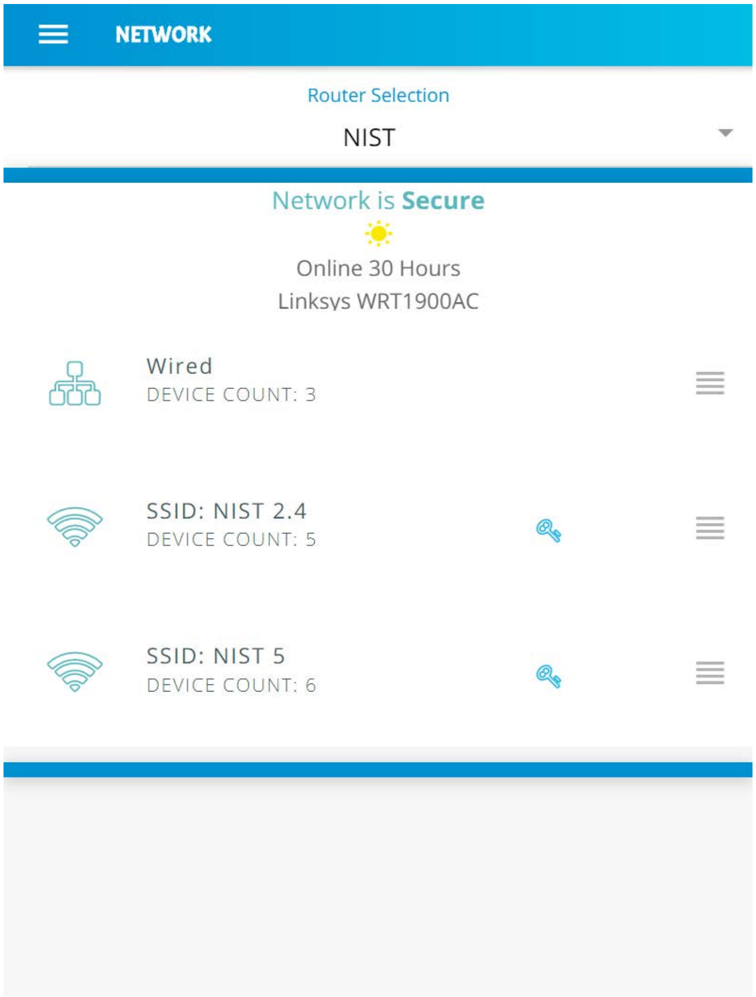
yikes!

Automated Network Security

Jane.Doe@gmail.com

•••••••••••••••

5. The home screen will show the network overview:



Router Selection

NIST

Network is **Secure**

Online 30 Hours

Linksys WRT1900AC

Wired
DEVICE COUNT: 3

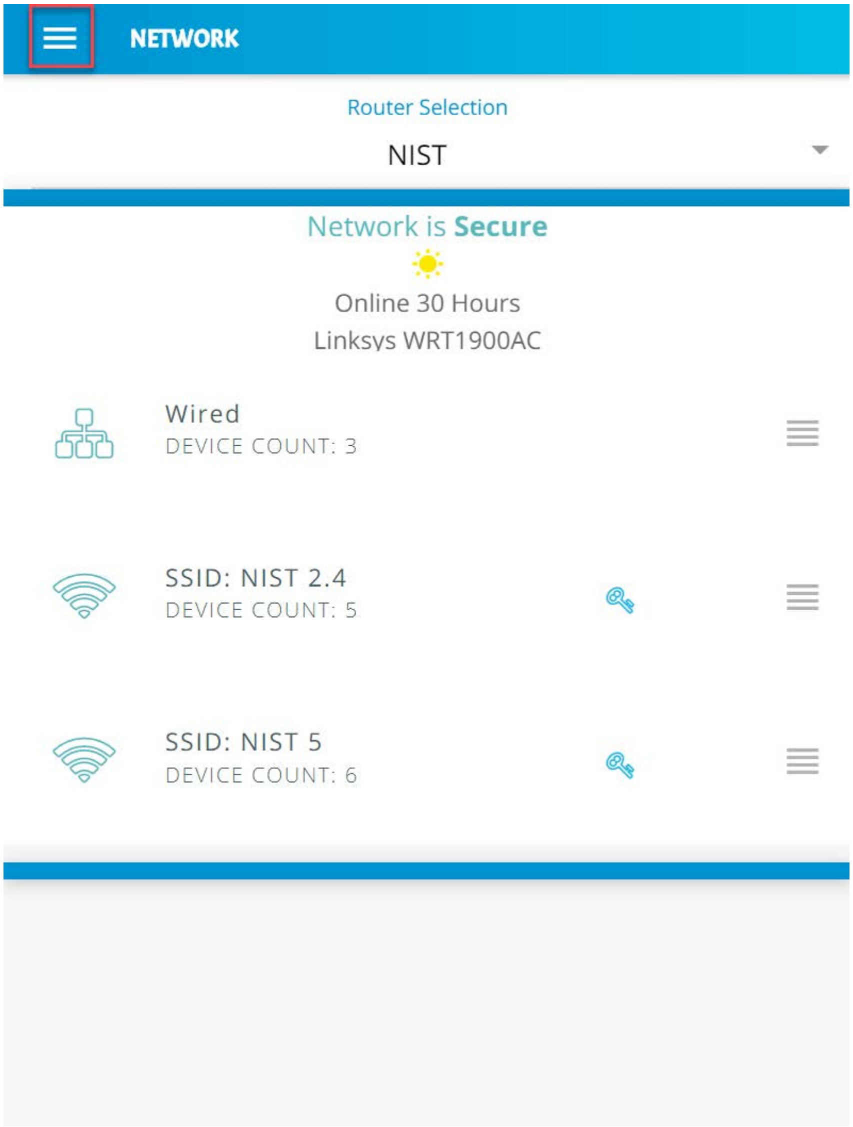
SSID: NIST 2.4
DEVICE COUNT: 5

SSID: NIST 5
DEVICE COUNT: 6

3.9.3.2 *Yikes! MUD-Capable IoT Device Discovery*

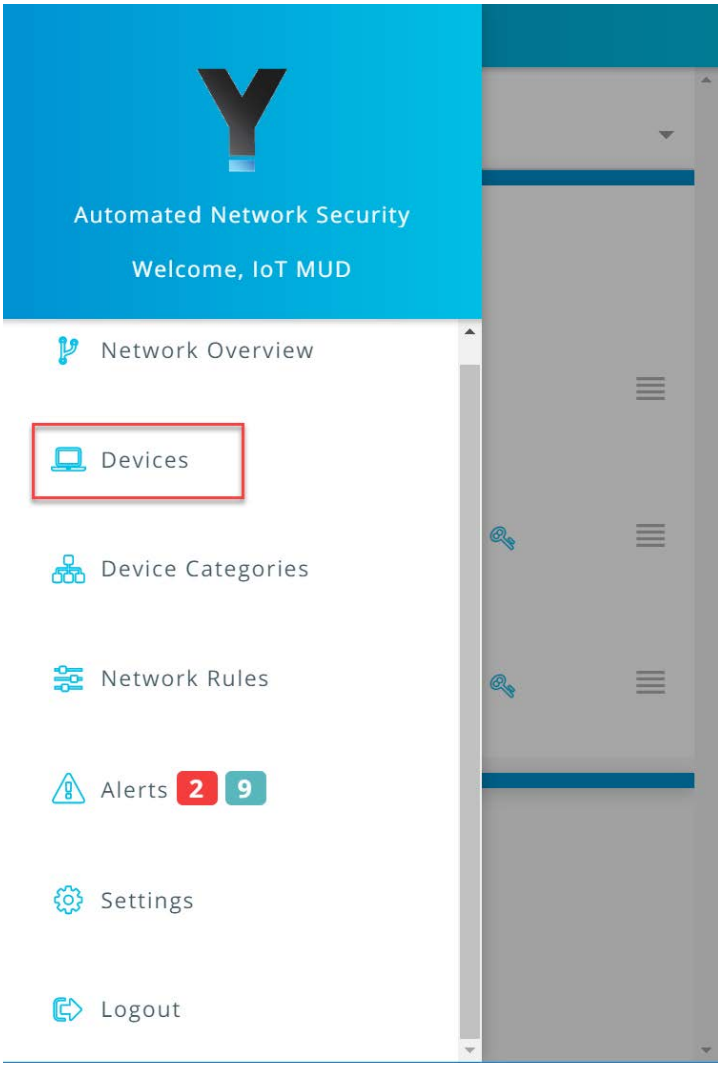
This section details the Yikes! MUD-capable IoT device discovery capability. This feature is accessible through the Yikes! mobile application and identifies all MUD-capable IoT devices that are connected to the network.

1549 1. Open the menu pane in the UI:



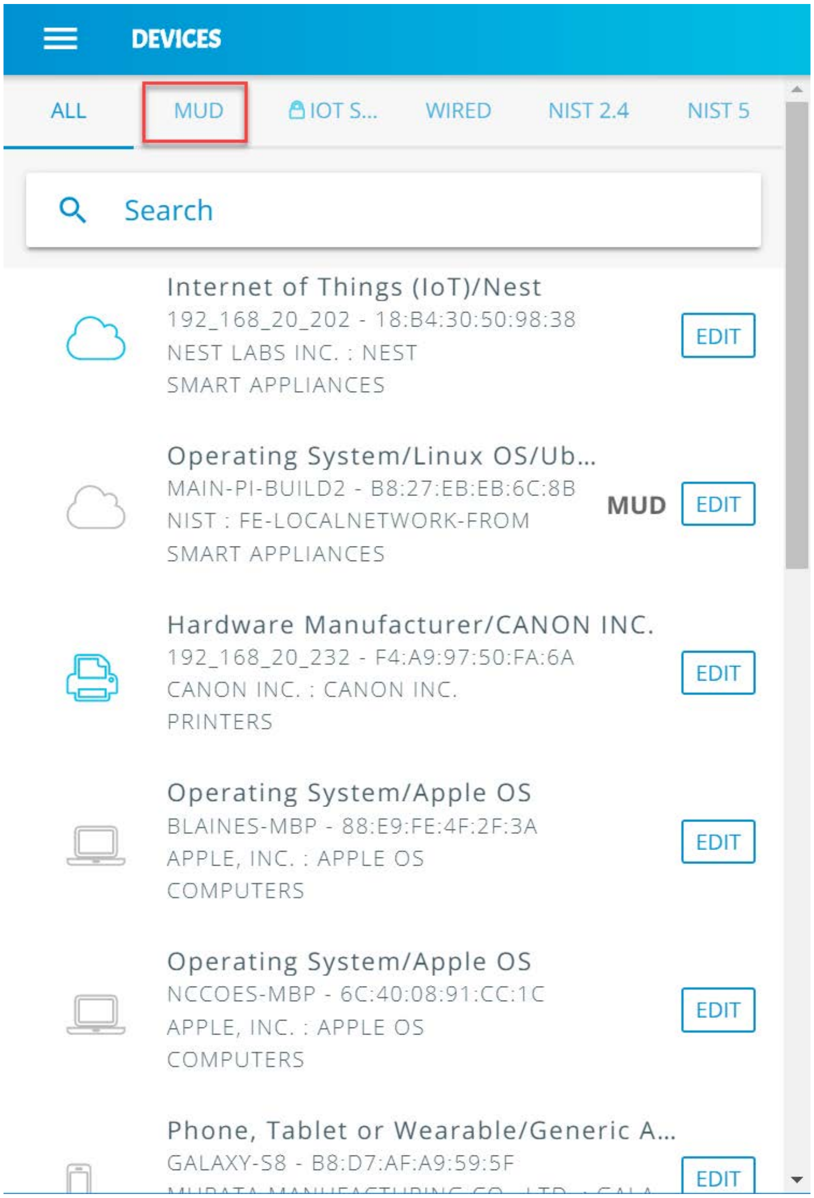
1550

1551 2. Click the **Devices** button to open the devices menu:



1552

3. Click the **MUD** tab to switch from the **ALL** device view to review the MUD-capable IoT devices connected to the network:



Internet of Things (IoT)/Nest
192_168_20_202 - 18:B4:30:50:98:38
NEST LABS INC. : NEST
SMART APPLIANCES

Operating System/Linux OS/Ub...
MAIN-PI-BUILD2 - B8:27:EB:EB:6C:8B
NIST : FE-LOCALNETWORK-FROM
SMART APPLIANCES

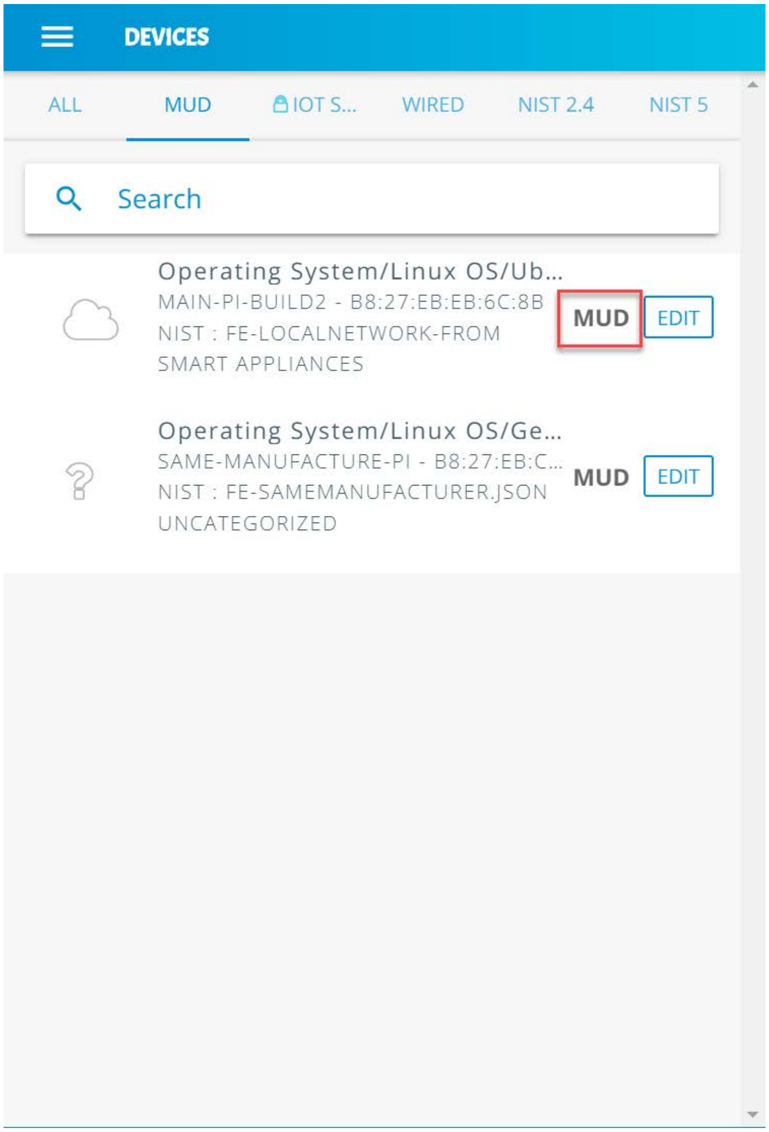
Hardware Manufacturer/CANON INC.
192_168_20_232 - F4:A9:97:50:FA:6A
CANON INC. : CANON INC.
PRINTERS

Operating System/Apple OS
BLAINES-MBP - 88:E9:FE:4F:2F:3A
APPLE, INC. : APPLE OS
COMPUTERS

Operating System/Apple OS
NCCOES-MBP - 6C:40:08:91:CC:1C
APPLE, INC. : APPLE OS
COMPUTERS

Phone, Tablet or Wearable/Generic A...
GALAXY-S8 - B8:D7:AF:A9:59:5F

4. All MUD-capable devices on the network will have the **MUD** label as seen below:



Operating System/Linux OS/Ub...
MAIN-PI-BUILD2 - B8:27:EB:EB:6C:8B
NIST : FE-LOCALNETWORK-FROM
SMART APPLIANCES
MUD EDIT

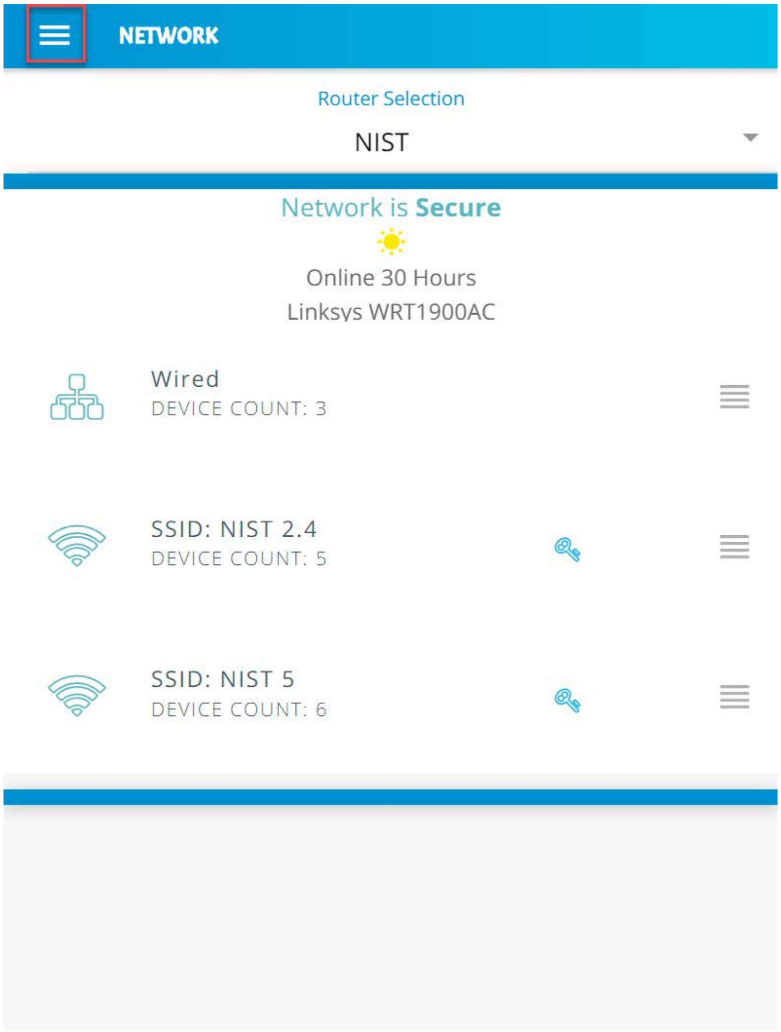
Operating System/Linux OS/Ge...
SAME-MANUFACTURE-PI - B8:27:EB:C...
NIST : FE-SAMEMANUFACTURER.JSON
UNCATEGORIZED
MUD EDIT

3.9.3.3 Yikes! Alerts

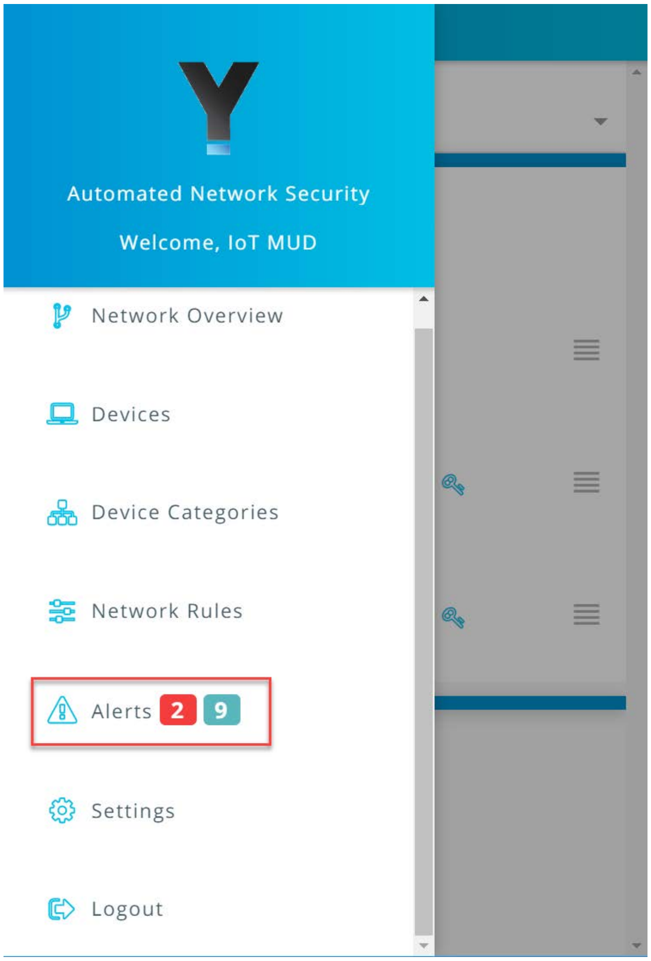
This section details the Yikes! alerting capability. This feature is accessible through the Yikes! mobile application and notifies users when new devices have been connected to the network. Additionally, this feature alerts the user when new devices are not recognized as known devices and are placed in the uncategorized device category by the Yikes! cloud.

From the Yikes! mobile application, the user can edit the information about the device (e.g., name, make, and model) and modify the device's category or can choose to ignore the alert by removing the notification.

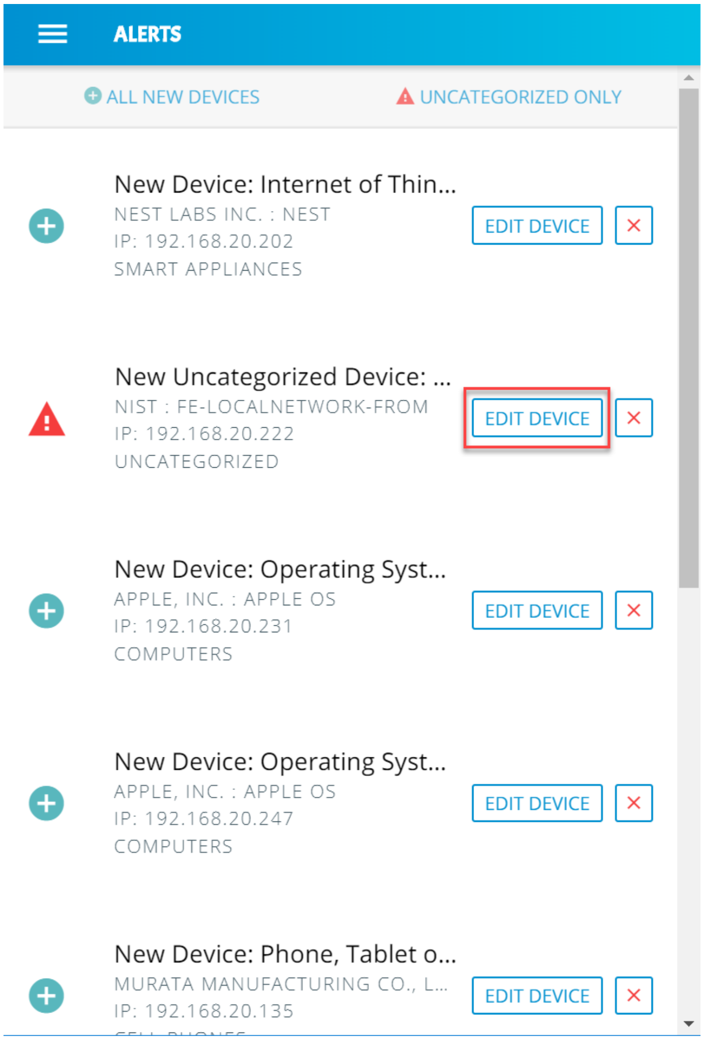
1. Open the menu pane in the UI:



1568 2. Click **Alerts** to open the Alerts menu:

1569 

3. Select a device to edit the device information and category by clicking **Edit Device:**



New Device: Internet of Thin...
NEST LABS INC. : NEST
IP: 192.168.20.202
SMART APPLIANCES

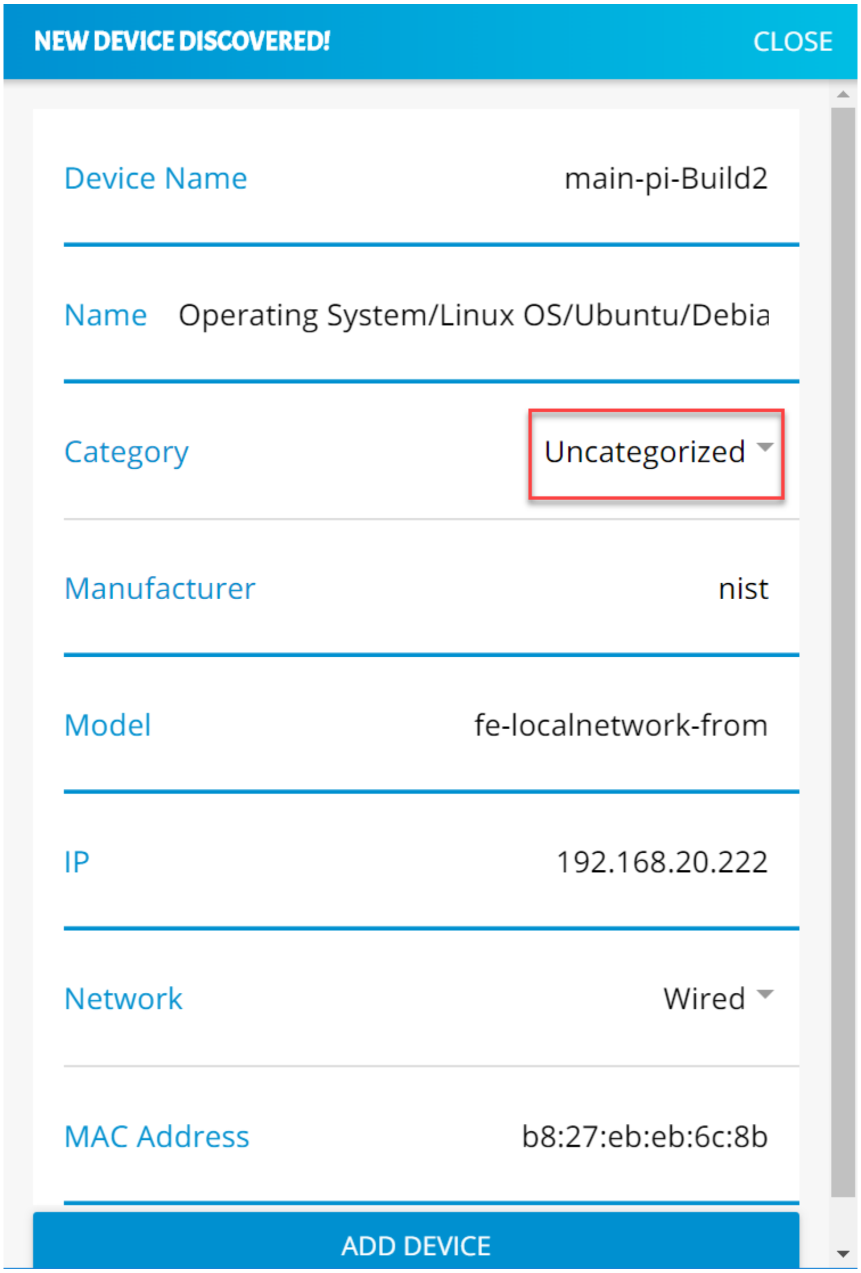
New Uncategorized Device: ...
NIST : FE-LOCALNETWORK-FROM
IP: 192.168.20.222
UNCATEGORIZED

New Device: Operating Syst...
APPLE, INC. : APPLE OS
IP: 192.168.20.231
COMPUTERS

New Device: Operating Syst...
APPLE, INC. : APPLE OS
IP: 192.168.20.247
COMPUTERS

New Device: Phone, Tablet o...
MURATA MANUFACTURING CO., L...
IP: 192.168.20.135

4. Modify the **Category** of the device by clicking the device’s current category:



Device Name main-pi-Build2

Name Operating System/Linux OS/Ubuntu/Debia

Category Uncategorized

Manufacturer nist

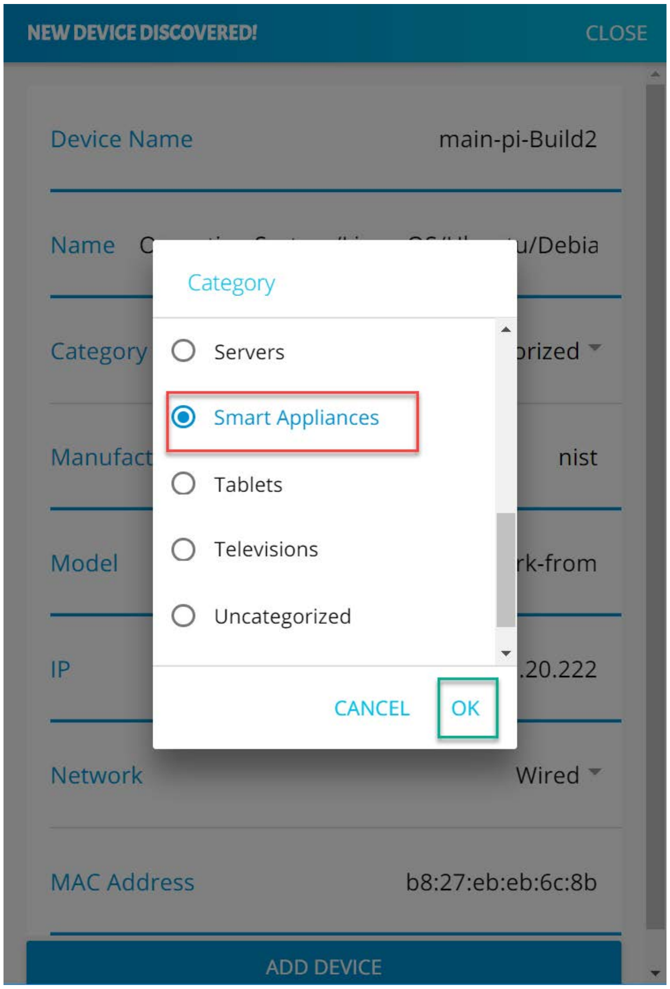
Model fe-localnetwork-from

IP 192.168.20.222

Network Wired

MAC Address b8:27:eb:eb:6c:8b

5. Select the desired category, in this case **Smart Appliances,** and click **OK:**



NEW DEVICE DISCOVERED!

Device Name main-pi-Build2

Name O u/Debia

Category orized

Manufact nist

Model rk-from

IP .20.222

Network Wired

MAC Address b8:27:eb:eb:6c:8b

Category

○ Servers

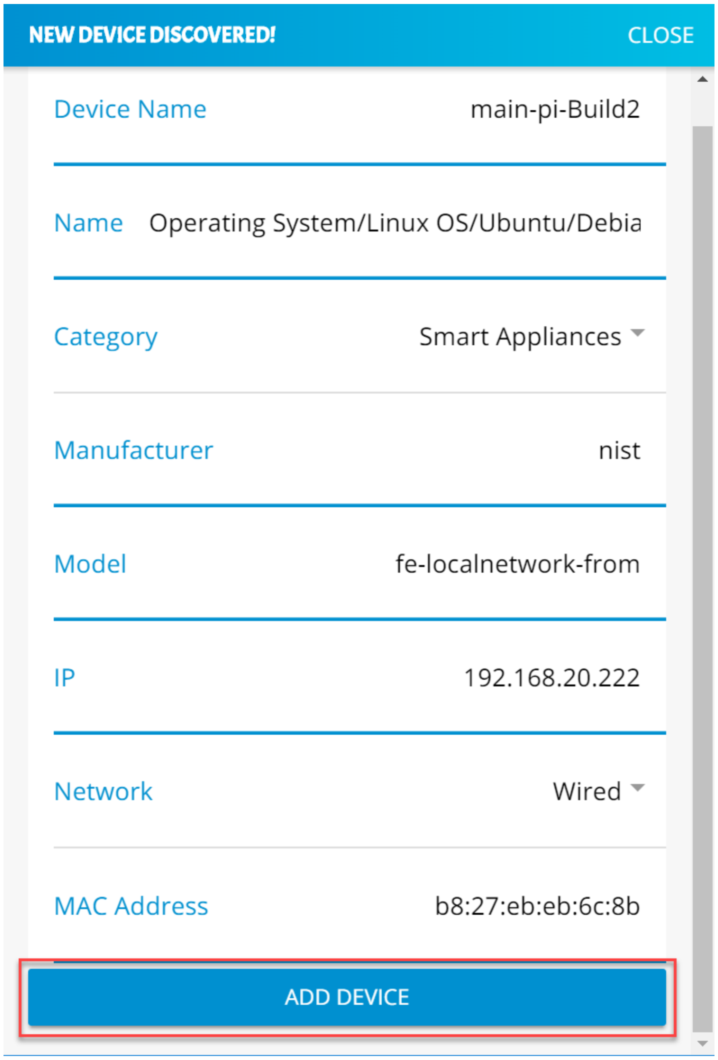
● Smart Appliances

○ Tablets

○ Televisions

○ Uncategorized

1576 6. The device **Category** will update to reflect the new selection. Click **Add Device** to complete the
1577 process:

1578 

NEW DEVICE DISCOVERED!

Device Name main-pi-Build2

Name Operating System/Linux OS/Ubuntu/Debia

Category Smart Appliances

Manufacturer nist

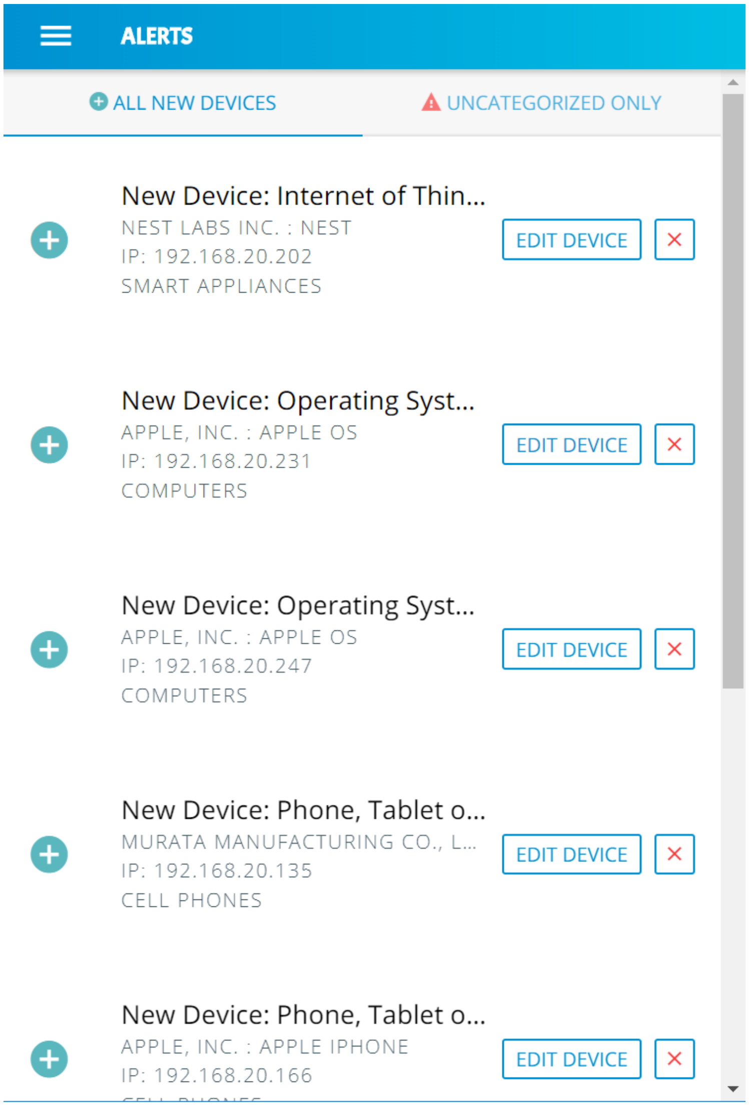
Model fe-localnetwork-from

IP 192.168.20.222

Network Wired

MAC Address b8:27:eb:eb:6c:8b

7. The alerts menu will update and no longer include the device that was just modified and added:



New Device: Internet of Thin...
NEST LABS INC. : NEST
IP: 192.168.20.202
SMART APPLIANCES

New Device: Operating Syst...
APPLE, INC. : APPLE OS
IP: 192.168.20.231
COMPUTERS

New Device: Operating Syst...
APPLE, INC. : APPLE OS
IP: 192.168.20.247
COMPUTERS

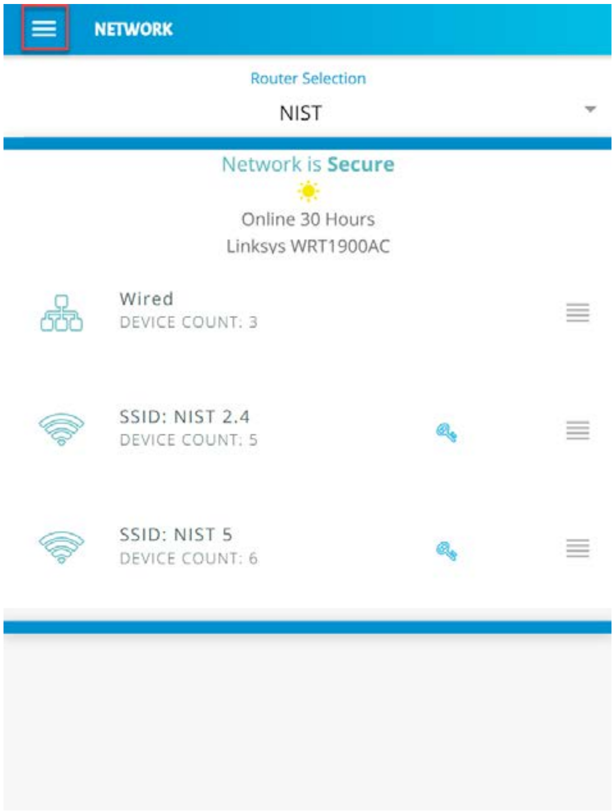
New Device: Phone, Tablet o...
MURATA MANUFACTURING CO., L...
IP: 192.168.20.135
CELL PHONES

New Device: Phone, Tablet o...
APPLE, INC. : APPLE IPHONE
IP: 192.168.20.166

3.9.3.4 Yikes! Device Categories and Setting Rules

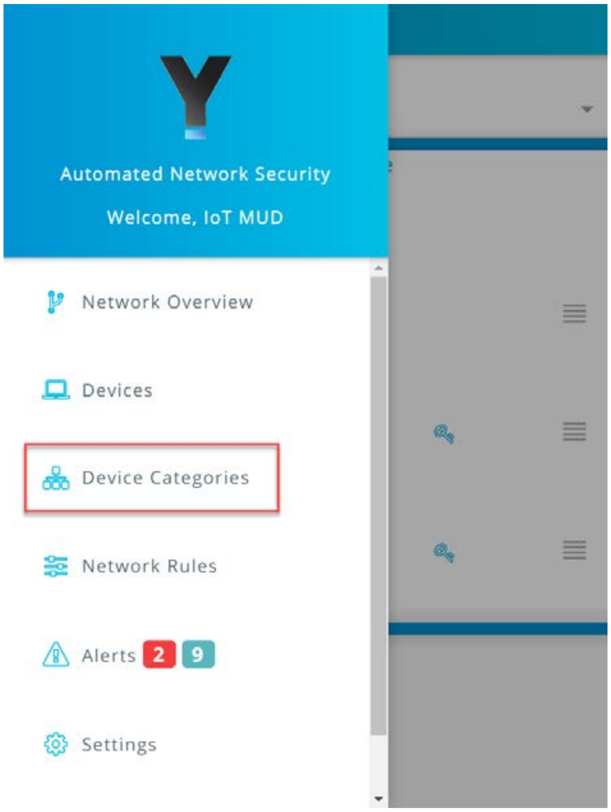
The Yikes! mobile application provides the capability to view predefined device categories and set rules for local communication between categories of devices on the local network and internet rules for all devices in a selected category.

1585 1. Click the menu bar to open the menu pane:



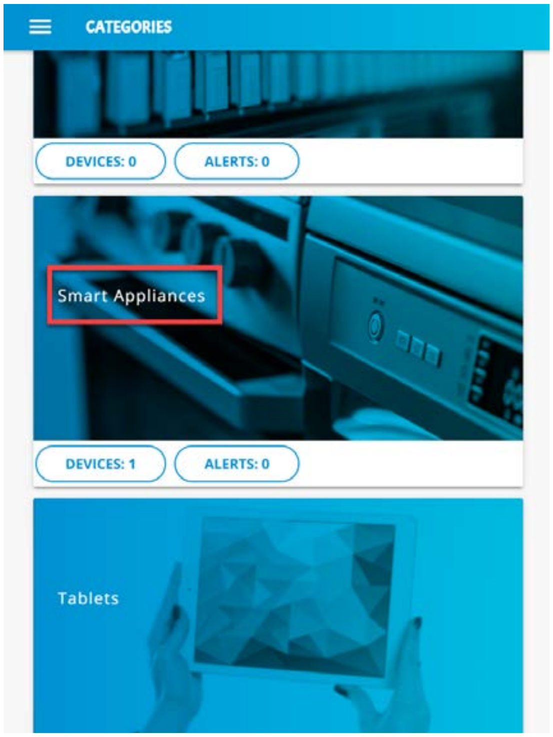
1586

1587 2. Click the **Device Categories** option to view all device categories:



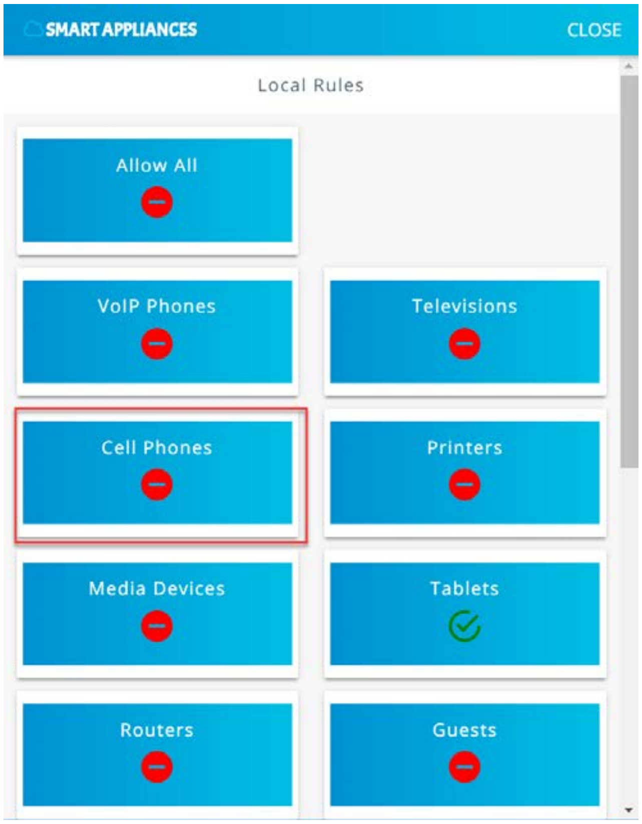
1588

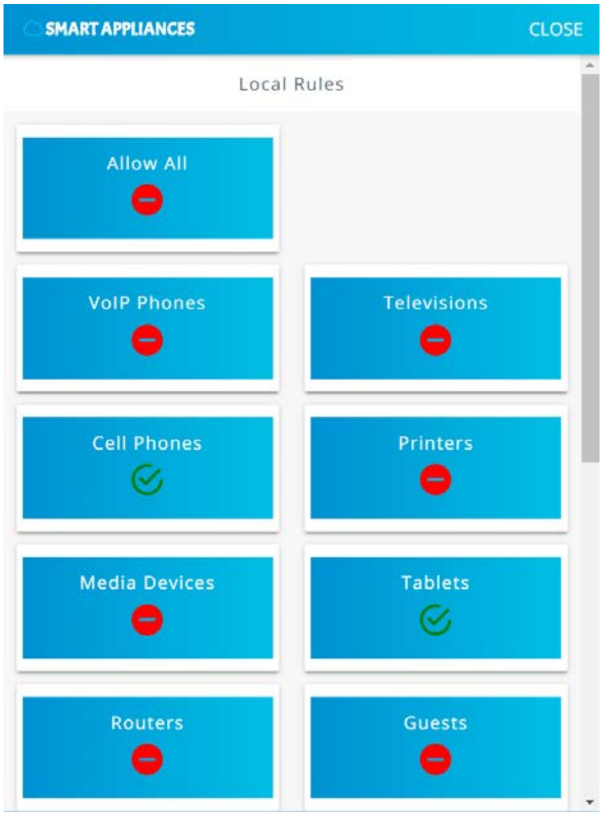
1589 3. Select the category of device to view and configure rules:



1590

4. Modify local rules by clicking on the category of devices with which the selected category is permitted to communicate:





SMART APPLIANCES CLOSE

Local Rules

Allow All ⊖

VoIP Phones ⊖

Televisions ⊖

Cell Phones ✓

Printers ⊖

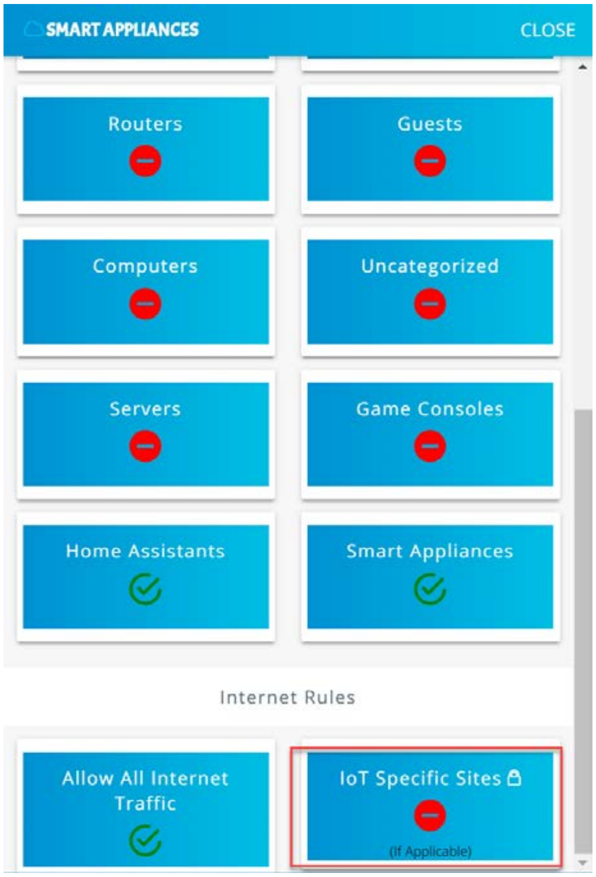
Media Devices ⊖

Tablets ✓

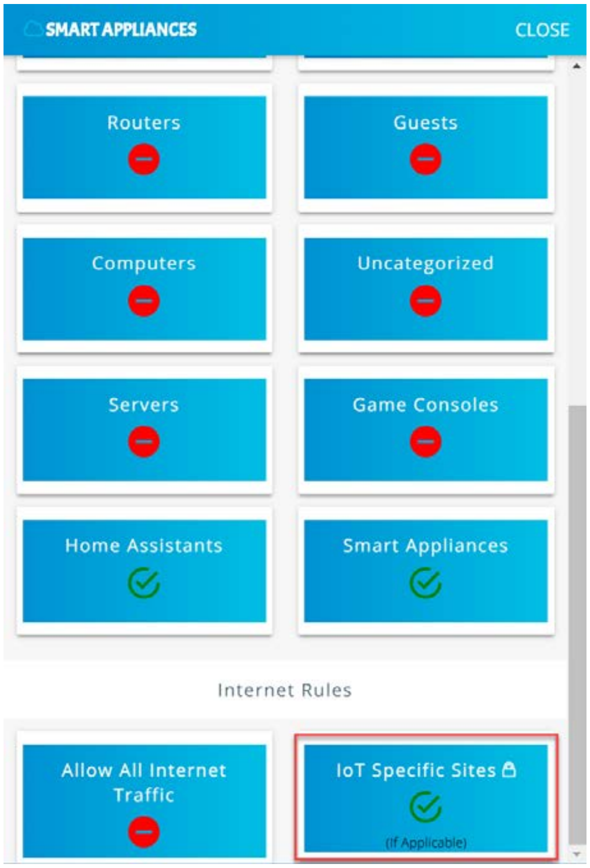
Routers ⊖

Guests ⊖

5. Scroll to the bottom of the page to view the current **Internet Rules** for this category, and change the permissions by clicking on **IoT Specific Sites:**

1597 

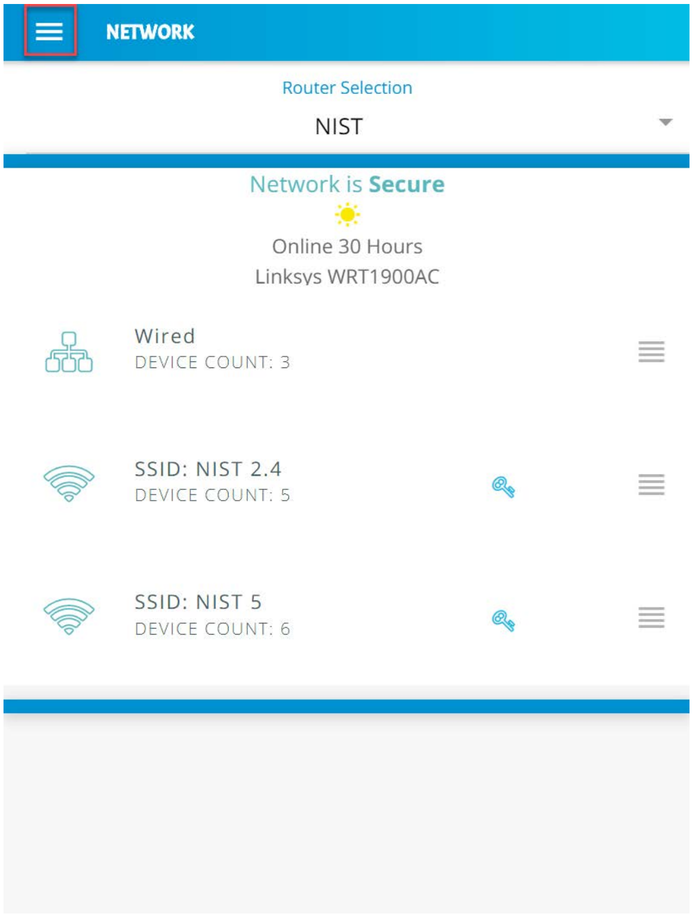
SMART APPLIANCES CLOSE



Smart appliances should now be permitted to communicate locally to Smart Appliances, Home Assistants, Tablets, Cell Phones, and, externally, to IoT Specific Sites.

3.9.3.5 Yikes! Network Rules

1. The Yikes! mobile application allows reviewing the rules that have been implemented on the network. These rules are divided into two main sections: Local Rules and Internet Rules. Local rules display the local communications permitted for each category of devices. Internet rules display the internet communications permitted for each category of devices. This section reviews the rules defined for Smart Appliances in Yikes! Device Categories and Setting Rules UI:

1607 

NETWORK

Router Selection

NIST

Network is **Secure**

Online 30 Hours

Linksys WRT1900AC

Wired

DEVICE COUNT: 3

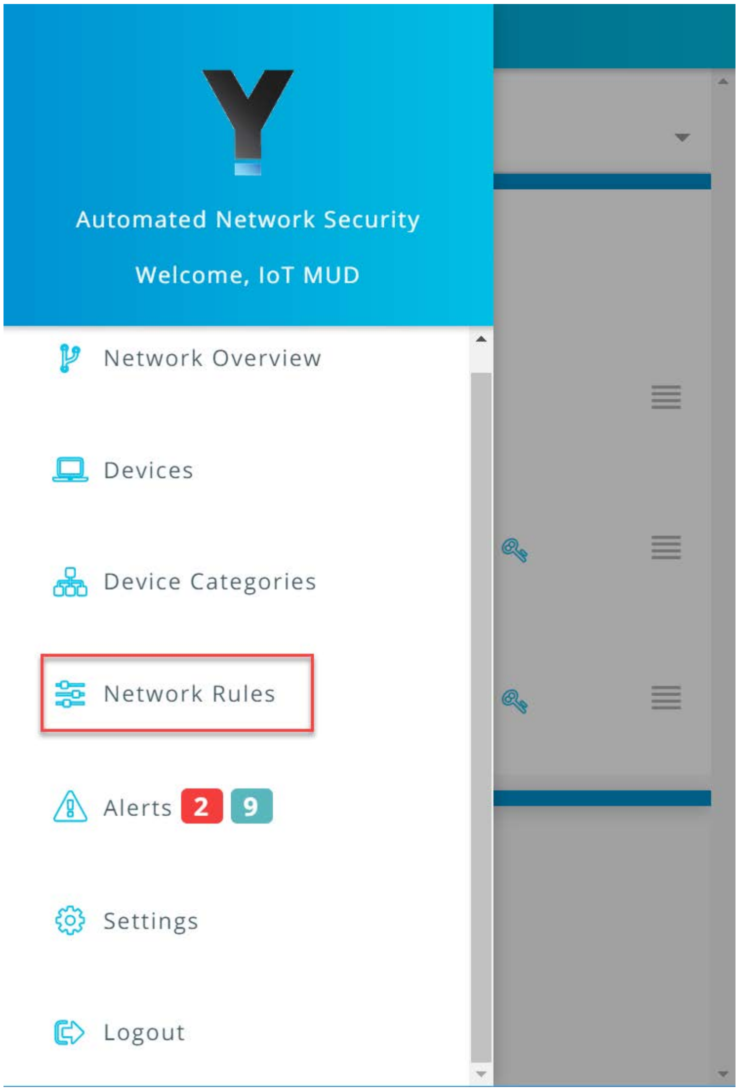
SSID: NIST 2.4

DEVICE COUNT: 5

SSID: NIST 5

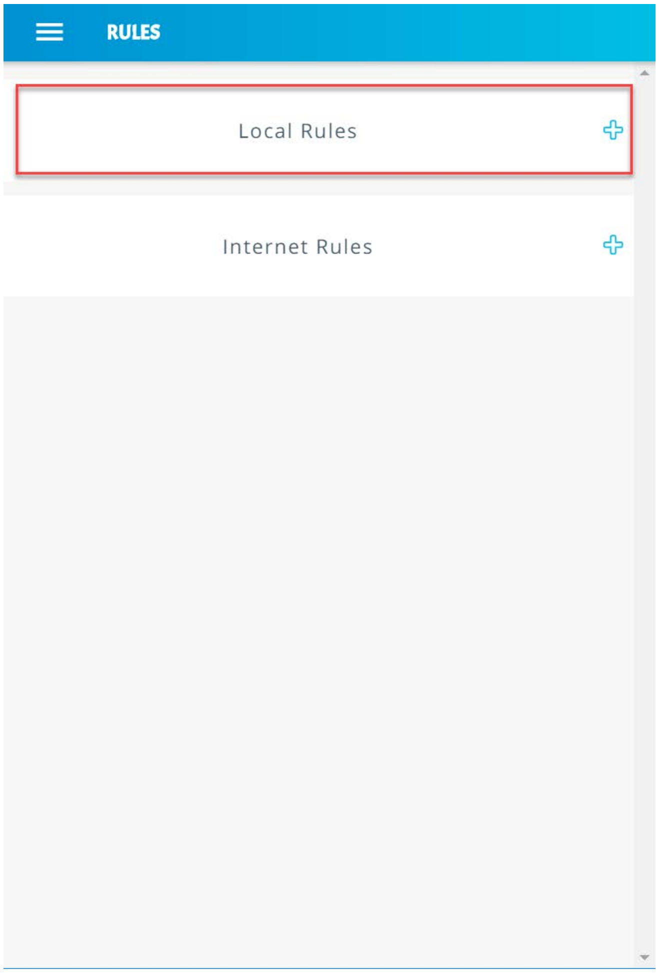
DEVICE COUNT: 6

1608 2. Click **Network Rules** to navigate to the rules menu:



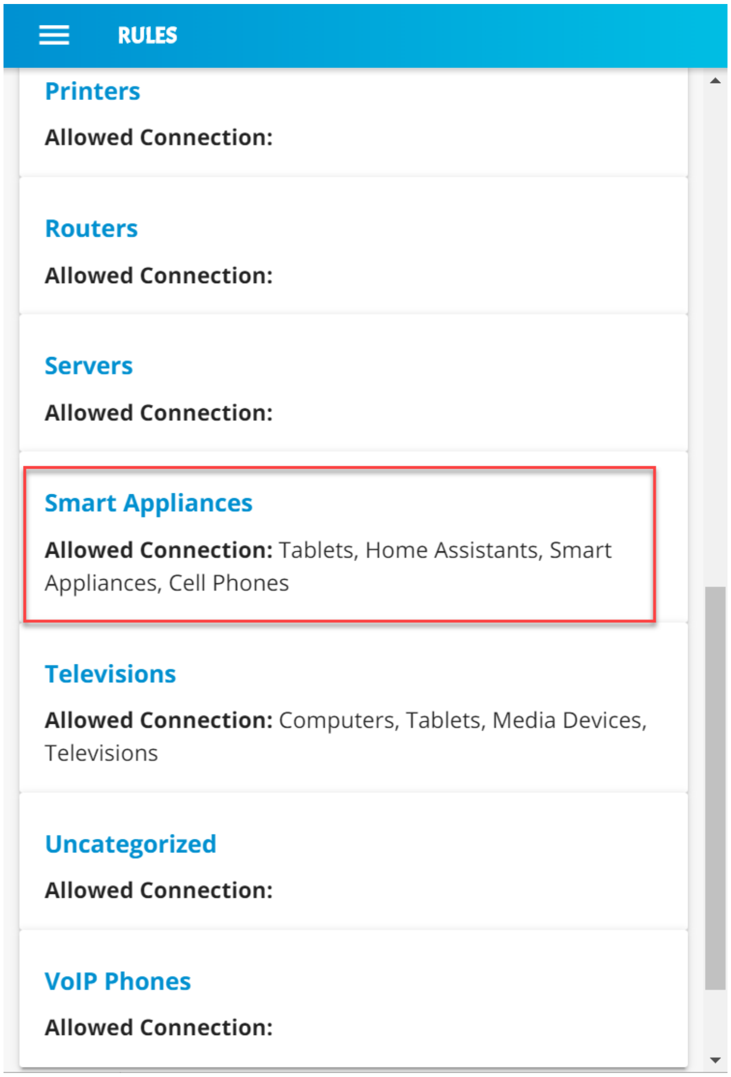
1609

1610 3. Click **Local Rules** to view the permitted local communications for each device category:



1611

1612 4. Scroll down to view the local rules for the **Smart Appliances** category:



RULES

Printers

Allowed Connection:

Routers

Allowed Connection:

Servers

Allowed Connection:

Smart Appliances

Allowed Connection: Tablets, Home Assistants, Smart Appliances, Cell Phones

Televisions

Allowed Connection: Computers, Tablets, Media Devices, Televisions

Uncategorized

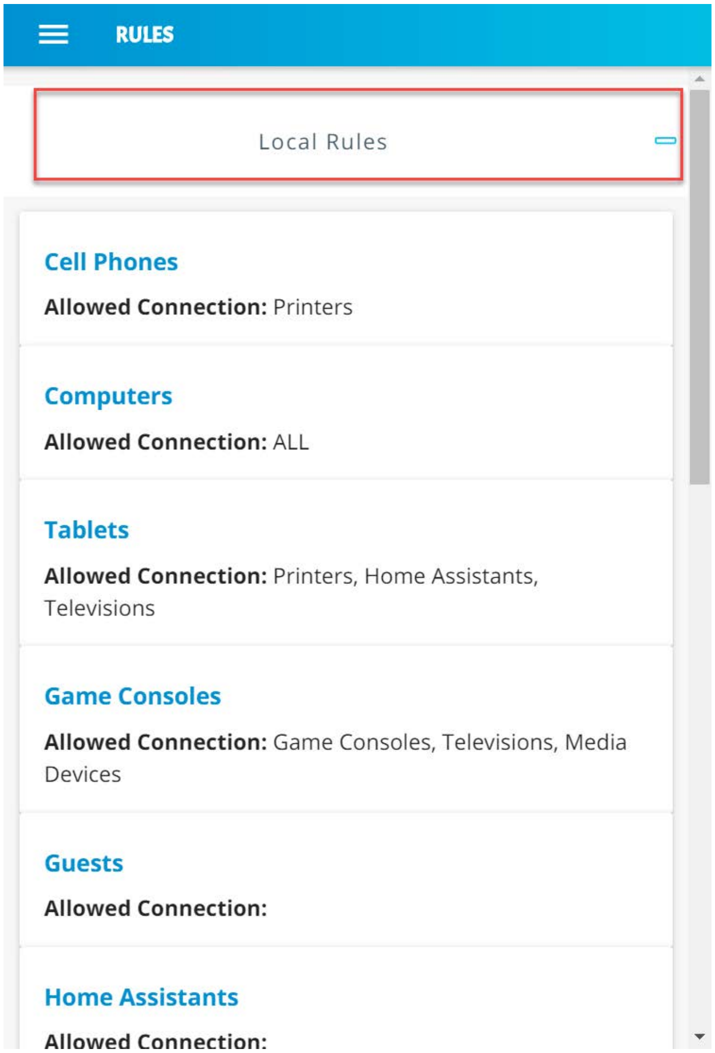
Allowed Connection:

VoIP Phones

Allowed Connection:

1613

1614 5. Minimize the rules by clicking the **Local Rules** button:



Cell Phones

Allowed Connection: Printers

Computers

Allowed Connection: ALL

Tablets

Allowed Connection: Printers, Home Assistants, Televisions

Game Consoles

Allowed Connection: Game Consoles, Televisions, Media Devices

Guests

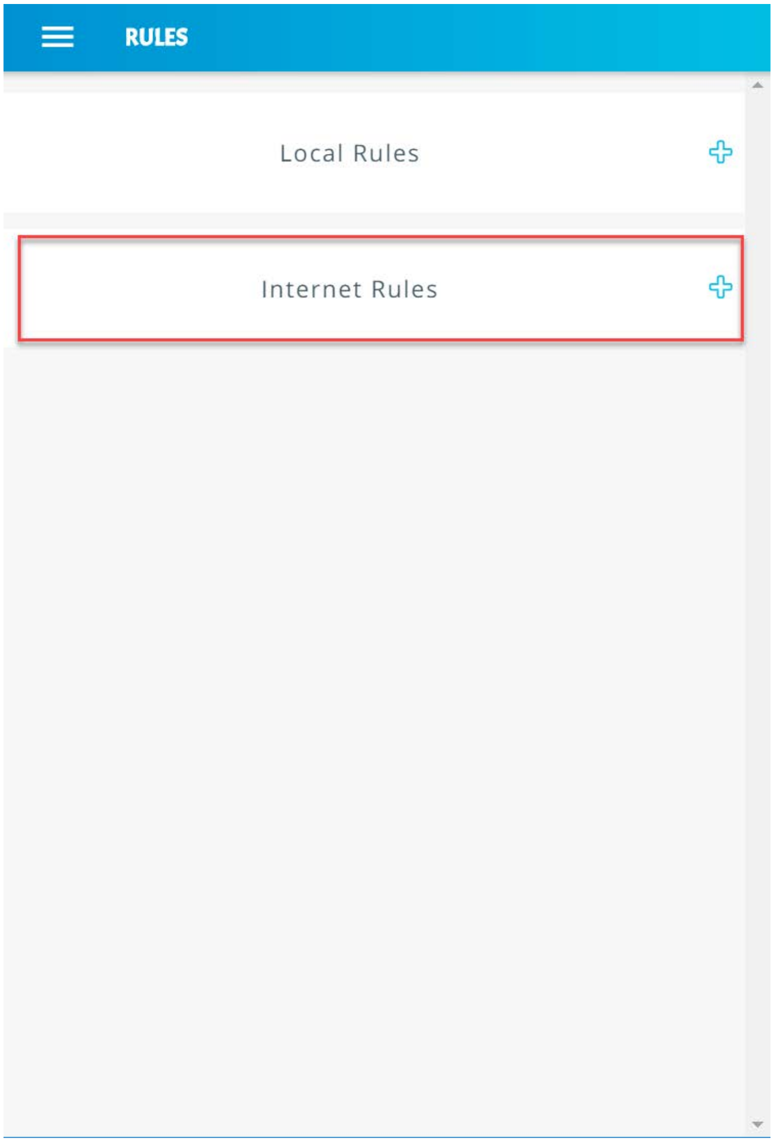
Allowed Connection:

Home Assistants

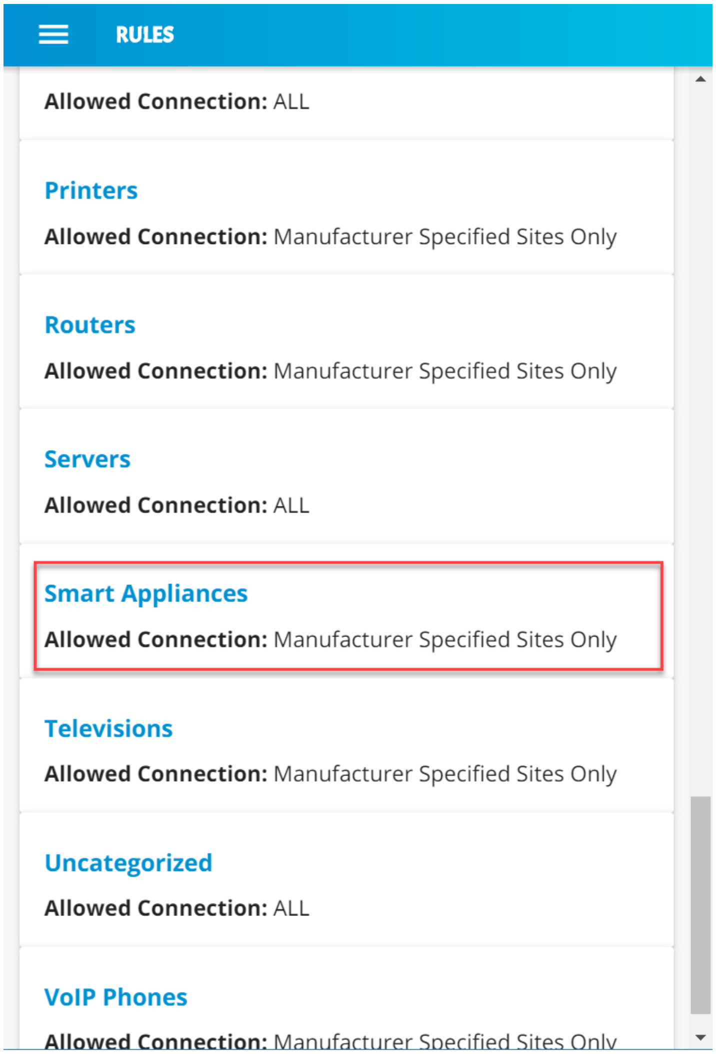
Allowed Connection:

1615

6. Expand the rules that show internet rules for device categories by clicking **Internet Rules:**



1618 7. Scroll down to view the internet rules for the **Smart Appliances** category:



RULES

Allowed Connection: ALL

Printers

Allowed Connection: Manufacturer Specified Sites Only

Routers

Allowed Connection: Manufacturer Specified Sites Only

Servers

Allowed Connection: ALL

Smart Appliances

Allowed Connection: Manufacturer Specified Sites Only

Televisions

Allowed Connection: Manufacturer Specified Sites Only

Uncategorized

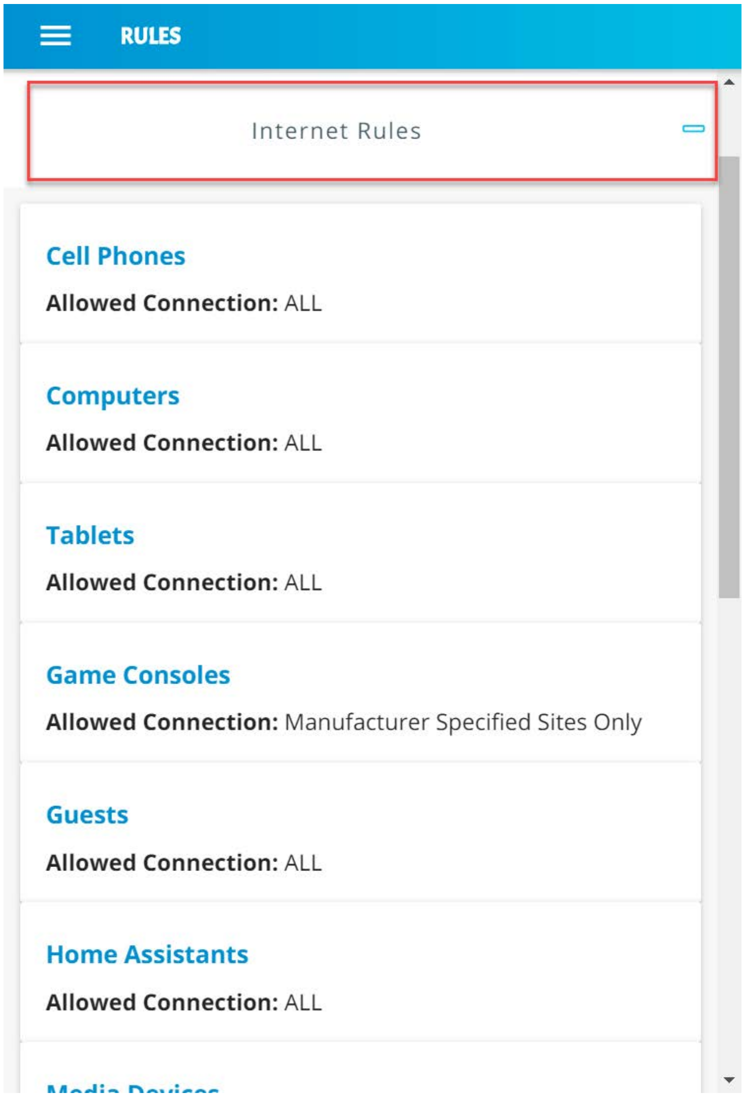
Allowed Connection: ALL

VoIP Phones

Allowed Connection: Manufacturer Specified Sites Only

1619

8. Minimize the rules by clicking the **Internet Rules** button:



RULES

Internet Rules

Cell Phones

Allowed Connection: ALL

Computers

Allowed Connection: ALL

Tablets

Allowed Connection: ALL

Game Consoles

Allowed Connection: Manufacturer Specified Sites Only

Guests

Allowed Connection: ALL

Home Assistants

Allowed Connection: ALL

Media Devices

3.10 GCA Quad9 Threat Signaling in Yikes! Router

This section describes the threat-signaling service provided by GCA in the Yikes! router. This capability should not require configuration because the Quad9 Active Threat Response (Q9Thrt) open-source software should be fully functional when the Yikes! router to connects to the network. Please see the Q9Thrt GitHub page for details on this software: https://github.com/osmud/q9thrt#q9thrt.

3.10.1 GCA Quad9 Threat Signaling in Yikes! Router Overview

The GCA Q9Thrt leverages DNS traffic by using Quad9 DNS services and threat intelligence from ThreatSTOP. As detailed in NIST SP 1800-15B, Q9Thrt is integrated into the Yikes! router and relies on the availability of three third-party services in the cloud: Quad9 DNS service, Quad9 threat API, and ThreatSTOP threat MUD file server. The Yikes! router is integrated with GCA Q9Thrt capabilities implemented, configured, and enabled out of the box.

3.10.2 Configuration Overview

At this implementation, no additional network, software, or hardware configuration was required to enable GCA Q9Thrt on the Yikes! router.

3.10.3 Setup

At this implementation, no additional setup was required to enable GCA Q9Thrt on the Yikes! router. See the Yikes! Router section for details on the router setup.

To take advantage of threat signaling, the Yikes! router uses the Quad9 DNS services for domain name resolution. GCA Quad threat signaling depends upon the Quad9 DNS services to be up and running. The Quad9 threat API must also be available to provide the Yikes! router with information regarding specific threats. In addition, for any given threat that is found, the MUD file server provided by the threat intelligence service that has flagged that threat as potentially dangerous must also be available. These are third-party services that GCA Q9Thrt relies upon to be set up, configured, and available.

It is possible to implement the Q9Thrt feature onto a non-Yikes! router. To integrate the Q9Thrt feature onto an existing router, see the open-source software on GitHub: https://github.com/osmud/q9thrt.

This software was designed for and has been integrated successfully using the OpenWRT platform but has the potential to be integrated into various networking environments. Instructions on how to deploy Q9thrt onto an existing router can be found on https://github.com/osmud/q9thrt#q9thrt.

4 Build 3 Product Installation Guides

This section of the practice guide contains detailed instructions for installing, configuring, and integrating the products used to implement Build 3. For additional details on Build 3's logical and physical architectures, please refer to NIST SP 1800-15B.

4.1 Product Installation

4.1.1 DigiCert Certificates

DigiCert's CertCentral web-based platform allows provisioning and management of publicly trusted X.509 certificates for a variety of purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. For Build 3, the Premium Certificate created in Build 1 was leveraged for signing the MUD files. Additionally, this implementation leveraged a standard SSL certificate to secure the cloud servers. You will need to request standard SSL certificates for each of the servers in your implementation. For this build we requested standard SSL certificates for two servers—the MUD file server and the Micronets service provider cloud server. To request and implement DigiCert certificates, follow the documentation in Build 1's DigiCert Certificates section and subsequent sections.

Once you have received the requested certificates, you can store these on the respective servers in your desired location. For this demonstration, we simply stored them in the workspace directory on the appropriate servers, but it is likely these would be stored in the /usr/lib or /etc/lib directories.

4.1.2 MUD Manager

This section describes the CableLabs MUD manager, which, for this implementation, is a cloud-provided service. This implementation leveraged the nccoe-build-3 branch of CableLabs MUD manager Git release. This service can be hosted by the implementer or another party. This documentation describes setting up your own MUD manager.

4.1.2.1 MUD Manager Overview

The CableLabs MUD manager is used by the Micronets Manager as a utility service to retrieve MUD files from a passed URL, parse the MUD file, and produce device communication restriction declarations that can be passed to the associated Micronets Gateway Service.

This Micronets MUD manager is hosted in the service provider cloud and for this implementation is on the same server as the other Micronets services. The MUD manager is responsible for retrieving MUD files and their associated signature files and executing verification as outlined in the MUD specification. It generates the ACLs for the device based on the MUD file and provides this information to the Micronets Manager.

4.1.2.2 Configuration Overview

The following subsections document the software and network configurations for the MUD manager. Please note that the MUD manager, Micronets Manager, Websocket Proxy, MUD registry, and MSO portal are all implemented on the same server, nccoe-server1.micronets.net.

4.1.2.2.1 Network Configuration

The nccoe-server1.micronets.net server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.2.2.2 Software Configuration

For this build, the server ran on an Ubuntu 18.04 LTS operating system. The MUD manager runs in its own docker container and is configured to use SSL/TLS encryption.

The following software is required to install, configure, and operate the MUD manager:

- an Ubuntu 18.04 LTS server reachable by the server hosting the Micronets Manager instances and any Micronets gateways
- docker (v18.06 or higher)
- curl
- NGINX

4.1.2.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the MUD manager:

- 4 gigabyte (GB) of RAM
- 50 GB of free disk space

4.1.2.3 Setup

The subsequent sections describe installing, configuring, and confirming general operation for the MUD manager.

4.1.2.3.1 Install and Set Up Dependencies

1. Make directory for downloading micronets-related scripts and packages:

   ```
   mkdir Projects/micronets/
   ```

2. Install **docker, curl,** and **NGINX** by entering the following command:

   ```
   sudo apt install docker curl nginx
   ```

3. Create an NGINX config file for this server (Note: If you are following the architecture for this implementation, all Micronets cloud components will be hosted on this server, and this will be the same config file that will be modified to add routes to the different Micronets services):

   sudo vim /etc/nginx/sites-available/<ServerURL>

   ```
   sudo vim /etc/nginx/sites-available/nccoe-server1.micronets.net
   ```

4. Add the following configuration block to the file and add the path to the certificate and key file received from your DigiCert standard SSL. (Note: Additional locations will be added to this configuration block as you continue to set up the different Micronets services.)

```
server {
        listen 443 ssl;
        listen [::]:443 ssl;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name nccoe-server1.micronets.net;

        location / {

                try_files $uri $uri/ =404;

        }

        ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-
        server1_micronets_net.crt;

        ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-
        server1_micronets_net.key;

}
```

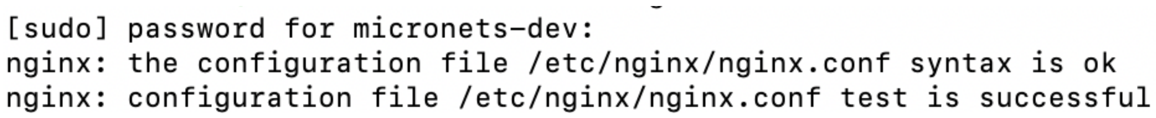
5. Enable the file by creating a link from it to the sites-enabled directory, which NGINX reads from during start-up:

```
sudo ln -s /etc/nginx/sites-available/nccoe-server1.micronets.net
/etc/nginx/sites-enabled/nccoe-server1.micronets.net
```

6. Next, test to make sure that there are no syntax errors in the NGINX files:

```
sudo nginx -t
```

You should see output similar to the following:



```
[sudo] password for micronets-dev:
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

7. If there are no problems, restart NGINX to enable your changes:

```
sudo systemctl restart nginx
```

4.1.2.3.2 Installing MUD Manager

1. Change directory to the Projects/micronets/ folder:

```
cd Projects/micronets/
```

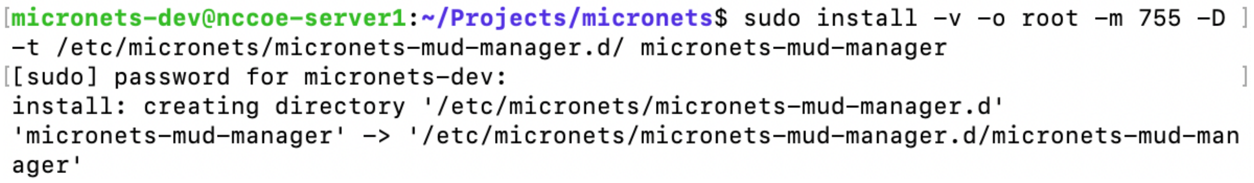
2. Download the management script by executing the following command:

```
curl -O https://raw.githubusercontent.com/cablelabs/micronets-mud-tools/nccoe-
build-3/bin/micronets-mud-manager
```

3. Install and execute the management script:

```
sudo install -v -o root -m 755 -D -t /etc/micronets/micronets-mud-manager.d/
micronets-mud-manager
```

You should see output similar to the following:



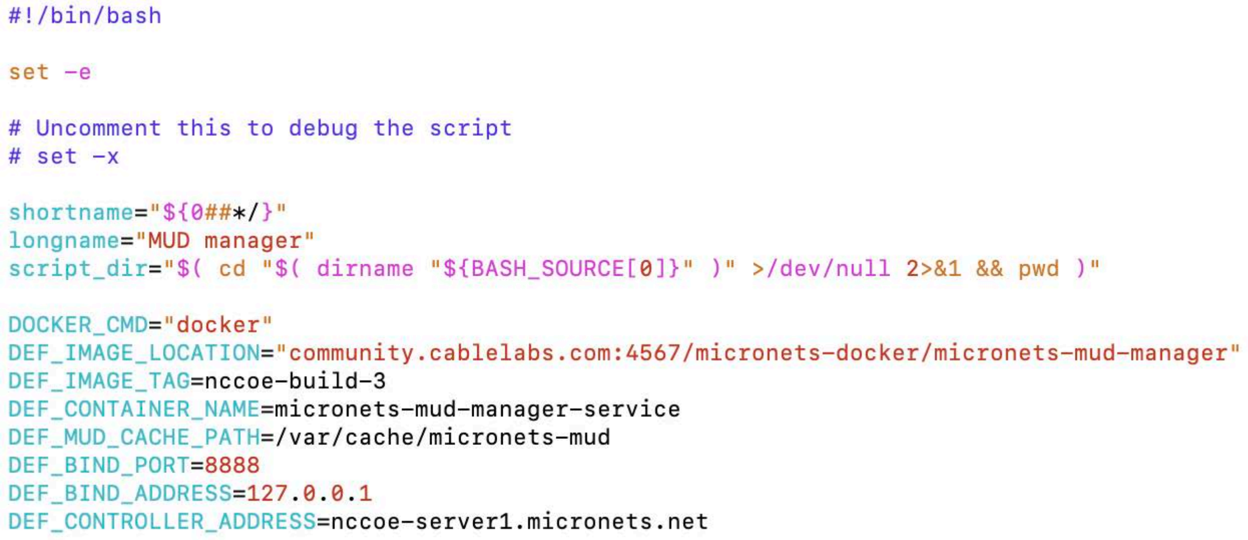
```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 -D ]
 -t /etc/micronets/micronets-mud-manager.d/ micronets-mud-manager
[[sudo] password for micronets-dev:                                                  ]
 install: creating directory '/etc/micronets/micronets-mud-manager.d'
 'micronets-mud-manager' -> '/etc/micronets/micronets-mud-manager.d/micronets-mud-man
 ager'
```

4. Open the management script to configure it for your implementation by entering the following command:

```
sudo vim /etc/micronets/micronets-mud-manager.d/micronets-mud-manager
```

5. Once the file is opened, modify the default variables in the management script to point to the server hosting our Micronets manager by changing the **DEF_CONTROLLER_ADDRESS** variable:

```
DEF_CONTROLLER_ADDRESS=nccoe-server1.micronets.net
```

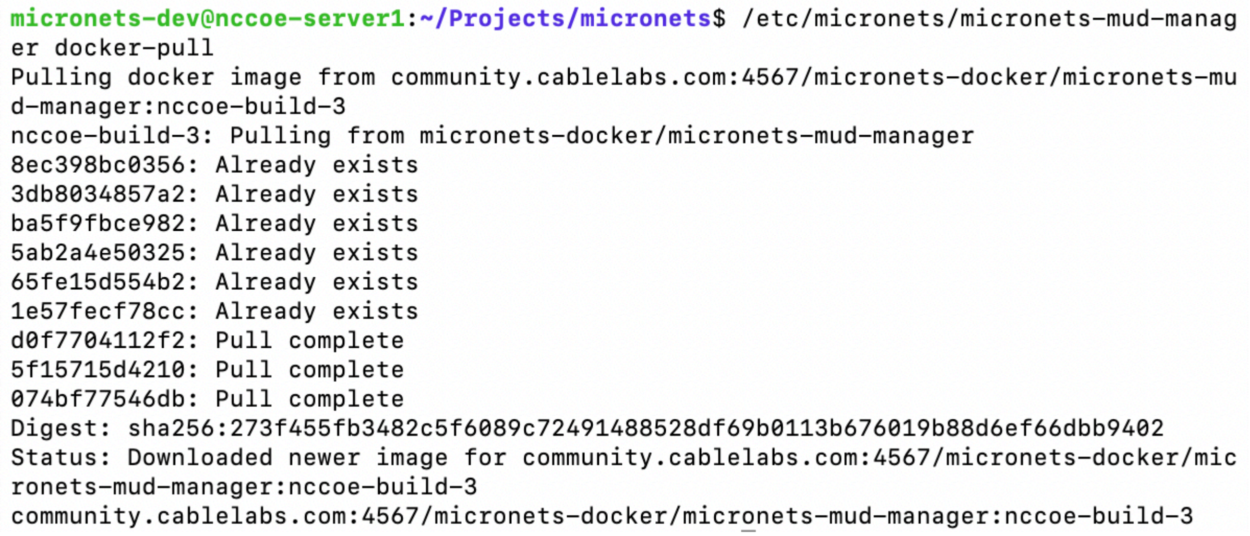


```
#!/bin/bash

set -e

# Uncomment this to debug the script
# set -x

shortname="${0##*/}"
longname="MUD manager"
script_dir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

DOCKER_CMD="docker"
DEF_IMAGE_LOCATION="community.cablelabs.com:4567/micronets-docker/micronets-mud-manager"
DEF_IMAGE_TAG=nccoe-build-3
DEF_CONTAINER_NAME=micronets-mud-manager-service
DEF_MUD_CACHE_PATH=/var/cache/micronets-mud
DEF_BIND_PORT=8888
DEF_BIND_ADDRESS=127.0.0.1
DEF_CONTROLLER_ADDRESS=nccoe-server1.micronets.net
```

6. Download the docker image by entering the following command:

   ```
   /etc/micronets/micronets-mud-manager.d/micronets-mud-manager docker-pull
   ```

   You should see output similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-mud-manag
er docker-pull
Pulling docker image from community.cablelabs.com:4567/micronets-docker/micronets-mu
d-manager:nccoe-build-3
nccoe-build-3: Pulling from micronets-docker/micronets-mud-manager
8ec398bc0356: Already exists
3db8034857a2: Already exists
ba5f9fbce982: Already exists
5ab2a4e50325: Already exists
65fe15d554b2: Already exists
1e57fecf78cc: Already exists
d0f7704112f2: Pull complete
5f15715d4210: Pull complete
074bf77546db: Pull complete
Digest: sha256:273f455fb3482c5f6089c72491488528df69b0113b676019b88d6ef66dbb9402
Status: Downloaded newer image for community.cablelabs.com:4567/micronets-docker/mic
ronets-mud-manager:nccoe-build-3
community.cablelabs.com:4567/micronets-docker/micronets-mud-manager:nccoe-build-3
```

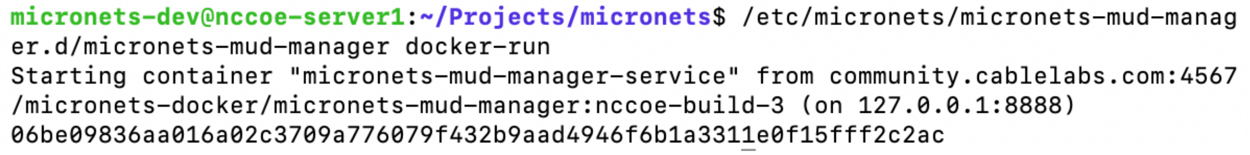
7. Next, set up the MUD cache directory by using the management script and entering the following command:

   ```
   sudo /etc/micronets/micronets-mud-manager.d/micronets-mud-manager setup-cache-
   dir
   ```

8. Last, start the MUD manager by entering the following command to run the docker container:

   ```
   /etc/micronets/micronets-mud-manager.d/micronets-mud-manager docker-run
   ```

   You should see output similar to the following:

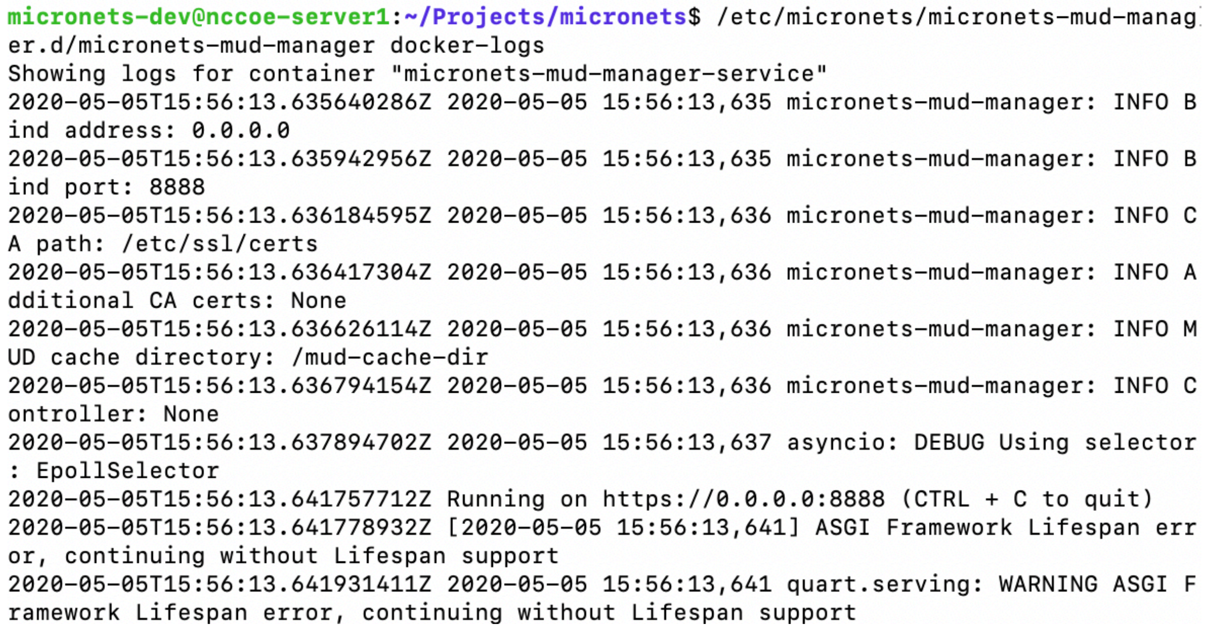


```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-mud-manag
er.d/micronets-mud-manager docker-run
Starting container "micronets-mud-manager-service" from community.cablelabs.com:4567
/micronets-docker/micronets-mud-manager:nccoe-build-3 (on 127.0.0.1:8888)
06be09836aa016a02c3709a776079f432b9aad4946f6b1a3311e0f15fff2c2ac
```

9. Verify that the MUD manager is running by using the following command and reviewing the logs:

   ```
   /etc/micronets/micronets-mud-manager.d/micronets-mud-manager docker-logs
   ```

   You should see output similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-mud-manag
er.d/micronets-mud-manager docker-logs
Showing logs for container "micronets-mud-manager-service"
2020-05-05T15:56:13.635640286Z 2020-05-05 15:56:13,635 micronets-mud-manager: INFO B
ind address: 0.0.0.0
2020-05-05T15:56:13.635942956Z 2020-05-05 15:56:13,635 micronets-mud-manager: INFO B
ind port: 8888
2020-05-05T15:56:13.636184595Z 2020-05-05 15:56:13,636 micronets-mud-manager: INFO C
A path: /etc/ssl/certs
2020-05-05T15:56:13.636417304Z 2020-05-05 15:56:13,636 micronets-mud-manager: INFO A
dditional CA certs: None
2020-05-05T15:56:13.636626114Z 2020-05-05 15:56:13,636 micronets-mud-manager: INFO M
UD cache directory: /mud-cache-dir
2020-05-05T15:56:13.636794154Z 2020-05-05 15:56:13,636 micronets-mud-manager: INFO C
ontroller: None
2020-05-05T15:56:13.637894702Z 2020-05-05 15:56:13,637 asyncio: DEBUG Using selector
: EpollSelector
2020-05-05T15:56:13.641757712Z Running on https://0.0.0.0:8888 (CTRL + C to quit)
2020-05-05T15:56:13.641778932Z [2020-05-05 15:56:13,641] ASGI Framework Lifespan err
or, continuing without Lifespan support
2020-05-05T15:56:13.641931411Z 2020-05-05 15:56:13,641 quart.serving: WARNING ASGI F
ramework Lifespan error, continuing without Lifespan support
```

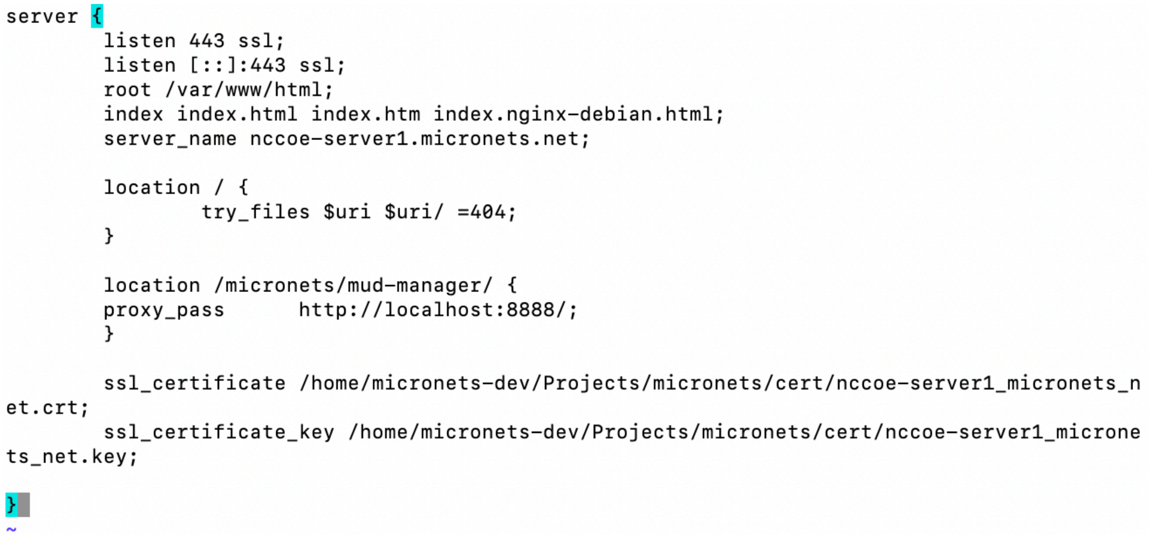
10. Set up a proxy pass to the MUD manager by adding the following entry to the NGINX `server` block:

    a. Open the NGINX sites-available file for the server:

    ```
    sudo vim /etc/nginx/sites-available/nccoe-server1.micronets.net
    ```

    b. Add the following location to the server block:

    ```
    location /micronets/mud-manager/ {
            proxy_pass     http://localhost:8888/;
    }
    ```



```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nccoe-server1.micronets.net;

        location / {
                try_files $uri $uri/ =404;
        }

        location /micronets/mud-manager/ {
        proxy_pass      http://localhost:8888/;
        }

        ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-server1_micronets_n
et.crt;
        ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-server1_microne
ts_net.key;

}
~
```

11. Reload the NGINX server by executing the following command:

```
sudo nginx -s reload
```

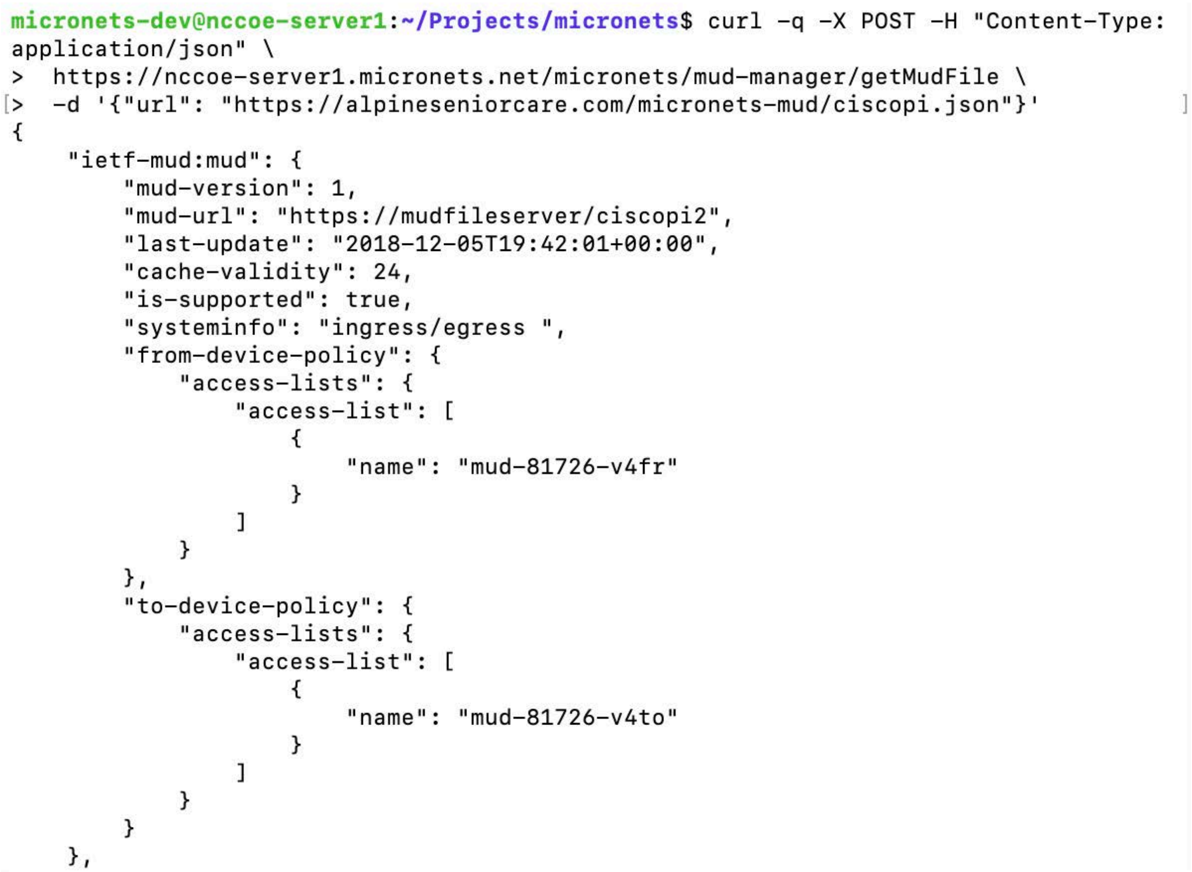
4.1.2.3.3 Operation

In this section, we test general operation of the MUD manager.

1. Test the MUD manager by retrieving a MUD file and using the following command (replace the MUD manager URL with the URL you created in Section 4.1.2.3.1):

```
curl -q -X POST -H "Content-Type: application/json" \
 https://nccoe-server1.micronets.net/micronets/mud-manager/getMudFile \
 -d '{"url": "https://alpineseniorcare.com/micronets-mud/ciscopi.json"}'
```

You should see the MUD file requested printed in the terminal:

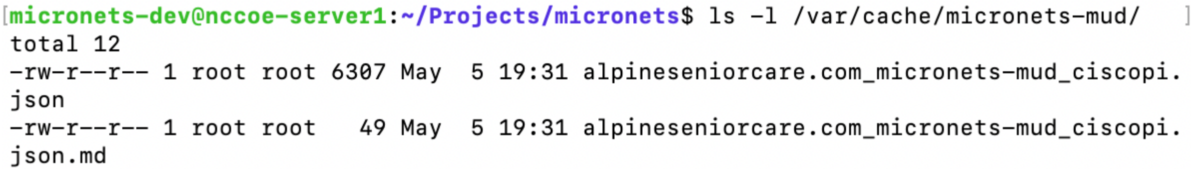


```
micronets-dev@nccoe-server1:~/Projects/micronets$ curl -q -X POST -H "Content-Type:
application/json" \
>  https://nccoe-server1.micronets.net/micronets/mud-manager/getMudFile \
[>  -d '{"url": "https://alpineseniorcare.com/micronets-mud/ciscopi.json"}'        ]
{
    "ietf-mud:mud": {
        "mud-version": 1,
        "mud-url": "https://mudfileserver/ciscopi2",
        "last-update": "2018-12-05T19:42:01+00:00",
        "cache-validity": 24,
        "is-supported": true,
        "systeminfo": "ingress/egress ",
        "from-device-policy": {
            "access-lists": {
                "access-list": [
                    {
                        "name": "mud-81726-v4fr"
                    }
                ]
            }
        },
        "to-device-policy": {
            "access-lists": {
                "access-list": [
                    {
                        "name": "mud-81726-v4to"
                    }
                ]
            }
        }
    },
```

2. Check the MUD file cache directory to confirm that the MUD file requested is stored in the cache:

   ```
   ls -1  /var/cache/micronets-mud/
   ```

   You should see the MUD file you just requested stored in the cache directory:

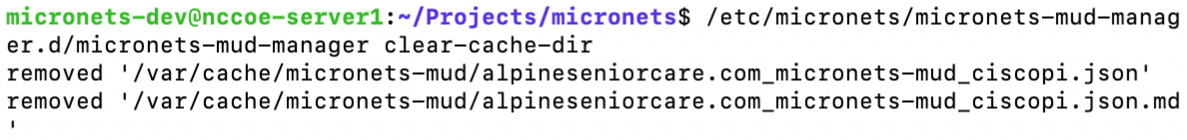


```
[micronets-dev@nccoe-server1:~/Projects/micronets$ ls -l /var/cache/micronets-mud/   ]
total 12
-rw-r--r-- 1 root root 6307 May  5 19:31 alpineseniorcare.com_micronets-mud_ciscopi.
json
-rw-r--r-- 1 root root   49 May  5 19:31 alpineseniorcare.com_micronets-mud_ciscopi.
json.md
```

3. Now that the MUD manager has successfully retrieved its first MUD file, you can clear the cache by entering the following command:

   ```
   /etc/micronets/micronets-mud-manager.d/micronets-mud-manager clear-cache-dir
   ```

   You should see the following output once the command above has been executed:

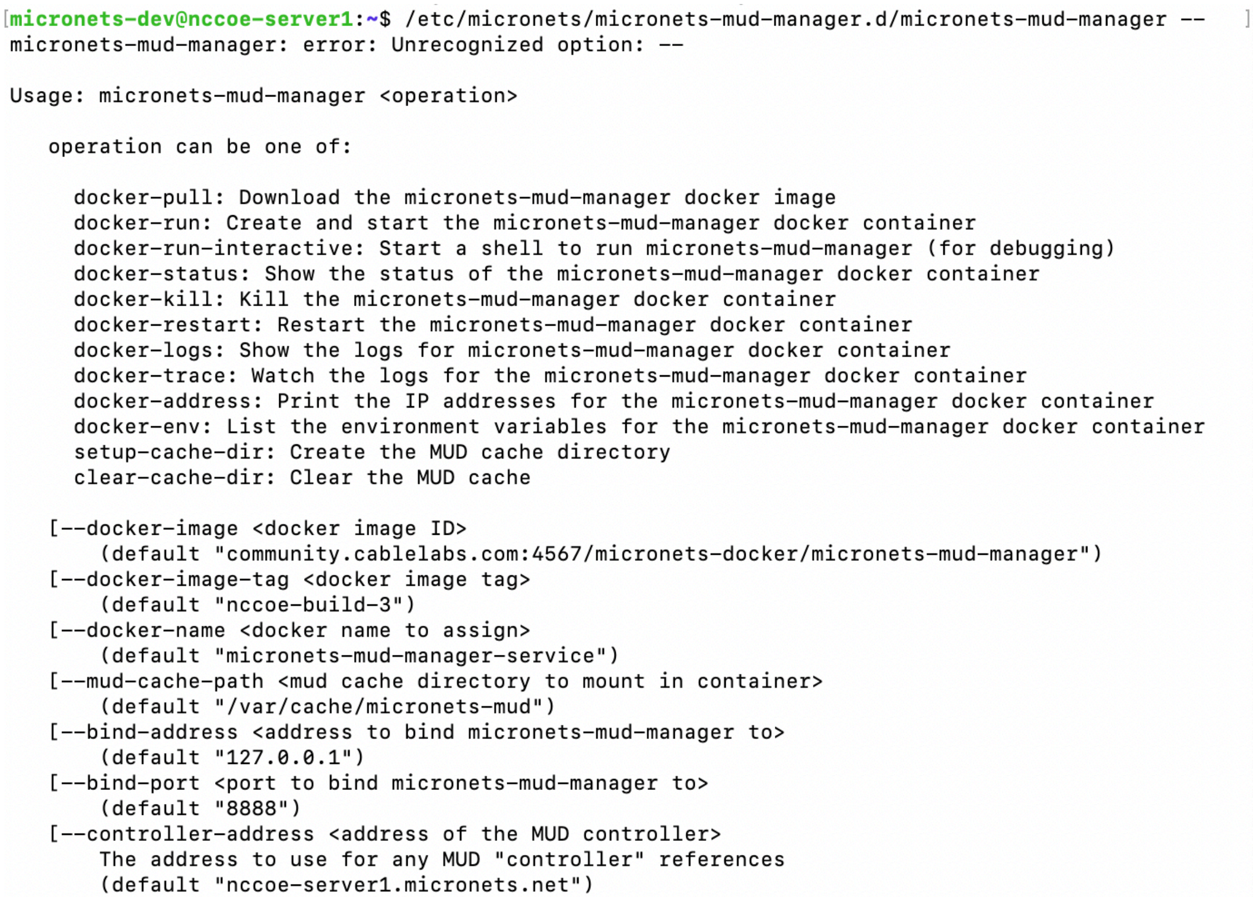


```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-mud-manag
er.d/micronets-mud-manager clear-cache-dir
removed '/var/cache/micronets-mud/alpineseniorcare.com_micronets-mud_ciscopi.json'
removed '/var/cache/micronets-mud/alpineseniorcare.com_micronets-mud_ciscopi.json.md
'
```

4. To output a list of additional docker commands supported by the management script, you can execute the following command:

```
/etc/micronets/micronets-mud-manager.d/micronets-mud-manager –
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ /etc/micronets/micronets-mud-manager.d/micronets-mud-manager --
micronets-mud-manager: error: Unrecognized option: --

Usage: micronets-mud-manager <operation>

   operation can be one of:

     docker-pull: Download the micronets-mud-manager docker image
     docker-run: Create and start the micronets-mud-manager docker container
     docker-run-interactive: Start a shell to run micronets-mud-manager (for debugging)
     docker-status: Show the status of the micronets-mud-manager docker container
     docker-kill: Kill the micronets-mud-manager docker container
     docker-restart: Restart the micronets-mud-manager docker container
     docker-logs: Show the logs for micronets-mud-manager docker container
     docker-trace: Watch the logs for the micronets-mud-manager docker container
     docker-address: Print the IP addresses for the micronets-mud-manager docker container
     docker-env: List the environment variables for the micronets-mud-manager docker container
     setup-cache-dir: Create the MUD cache directory
     clear-cache-dir: Clear the MUD cache

   [--docker-image <docker image ID>
       (default "community.cablelabs.com:4567/micronets-docker/micronets-mud-manager")
   [--docker-image-tag <docker image tag>
       (default "nccoe-build-3")
   [--docker-name <docker name to assign>
       (default "micronets-mud-manager-service")
   [--mud-cache-path <mud cache directory to mount in container>
       (default "/var/cache/micronets-mud")
   [--bind-address <address to bind micronets-mud-manager to>
       (default "127.0.0.1")
   [--bind-port <port to bind micronets-mud-manager to>
       (default "8888")
   [--controller-address <address of the MUD controller>
       The address to use for any MUD "controller" references
       (default "nccoe-server1.micronets.net")
```

4.1.3 MUD File Server

This section describes the CableLabs MUD file server, which is a cloud-hosted service. The Build 3 implementation is designed a bit differently from the other three builds insofar as it requires a MUD registry to be incorporated in the solution as described in Volume B. We describe the MUD registry in this section of the documentation.

4.1.3.1 MUD File Server Overview

In the absence of a commercial MUD file server for use in this project, the NCCoE leveraged a Linode cloud-hosted Linux server to create the MUD file server that is accessible via the internet. This file server stores the MUD files along with their corresponding signature files for the IoT devices used in the project. Upon receiving a GET request for the MUD files and signatures, it serves the request to the MUD manager by using https.

4.1.3.2 Configuration Overview

The following subsections document the software and network configurations for the MUD file server.

4.1.3.2.1 Network Configuration

This server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.3.2.2 Software Configuration

For this build, the server ran on an Ubuntu 18.04 LTS operating system. The MUD files and signatures were hosted by an NGINX web server and configured to use SSL/TLS encryption.

4.1.3.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the MUD file server:

- 4 GB of RAM
- 50 GB of free disk space

4.1.3.3 Setup

4.1.3.3.1 NGINX Web Server

1. Update your local package index by entering the following command:

   ```
   sudo apt update
   ```

2. Install NGINX by entering the following command:

   ```
   sudo apt install nginx
   ```

3. Create the directory where the MUD files will be stored on the MUD file server as follows:

   ```
   sudo mkdir -p /var/www/nccoe-server2.micronets.net/html/micronets-mud/
   ```

4. Create an NGINX config file for this server (Note: If you are following the architecture for this implementation, all Micronets cloud components will be hosted on this server, and this will be the same config file that will be modified to add routes to the different Micronets services):

   sudo vim /etc/nginx/sites-available/<ServerURL>

   Below is an example of this command:

   ```
   sudo vim /etc/nginx/sites-available/nccoe-server2.micronets.net
   ```

5. Add the following configuration block to the file (Note: Additional locations will be added to this configuration block as you continue to set up the different Micronets services):

   ```
   server {
       listen 443 ssl;
       listen [::]:443 ssl;
       root /var/www/nccoe-server2.micronets.net/html;
       index index.html index.htm index.nginx-debian.html;
       server_name nccoe-serve2.micronets.net;
        location / {
          # First attempt to serve request as file, then
          # as directory, then fall back to displaying a 404.
          try_files $uri $uri/ =404;
      }
      if ($scheme != "https") {
          return 301 https://$host$request_uri;
      }
      ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-
      server2_micronets_net.crt;
      ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-
      server2_micronets_net.key;


   include /etc/nginx/micronets-subscriber-forwards/*.conf;
   }
   ```

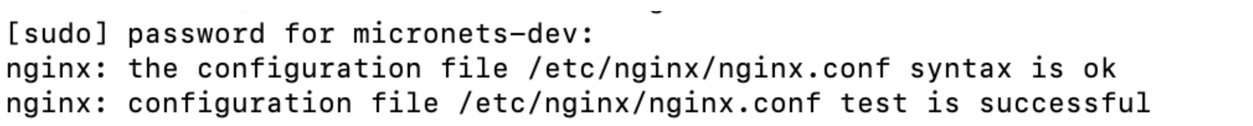
6. Enable the file by creating a link from it to the sites-enabled directory, which NGINX reads from during startup:

   ```
   sudo ln -s /etc/nginx/sites-available/nccoe-server2.micronets.net \
   /etc/nginx/sites-enabled/nccoe-server2.micronets.net
   ```

7. Next, test to make sure that there are no syntax errors in any of your NGINX files:

   ```
   sudo nginx -t
   ```

   You should see output similar to the following:

   ```
   [sudo] password for micronets-dev:
   nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
   nginx: configuration file /etc/nginx/nginx.conf test is successful
   ```

8. If there are no problems, restart NGINX to enable your changes:

   ```
   sudo systemctl restart nginx
   ```

4.1.3.3.2 MUD File Creation and Signing

To create MUD files for MUD-capable IoT devices, please follow the instructions in Build 1’s MUD File Server. Once MUD files and signature files are created, they can be stored in the web server directory created on the MUD file server in the previous section.

4.1.4 Micronets Gateway

This section describes the CableLabs Micronets Gateway, which, for this implementation, is an on-premise component. This implementation leveraged the nccoe-build-3 tagged version of CableLabs Micronets Gateway Git release. This documentation describes setting up your own Micronets gateway.

4.1.4.1 Micronets Gateway Overview

The Micronets Gateway establishes a connection to the Micronets Manager through the Websocket Proxy and receives traffic flow rules and other configuration information that it applies and enforces. Additionally, the Micronets Gateway supports wired and wireless connections, MUD-defined ACLs, and DPP onboarding.

4.1.4.2 Configuration Overview

The following subsections document the software and network configurations for the Micronets Gateway.

4.1.4.2.1 Network Configuration

Implementation of a Micronets gateway requires an internet source such as a digital subscriber line (DSL) or cable modem.

4.1.4.2.2 Software Configuration

The Micronets Gateway runs an Ubuntu 16.04 LTS server, which can support all the software dependencies and packages that will be installed during setup.

4.1.4.2.3 Hardware Configuration

For this implementation, we leveraged a Shuttle XPC slim DH170 with the following specs:

- x86_64 processor (Intel or AMD)
- at least two Ethernet ports
- wireless adapter with a QUALCOMM Atheros AR9271 chipset
- 2 GB or higher of RAM

4.1.4.3 Setup

4.1.4.3.1 Install Dependencies

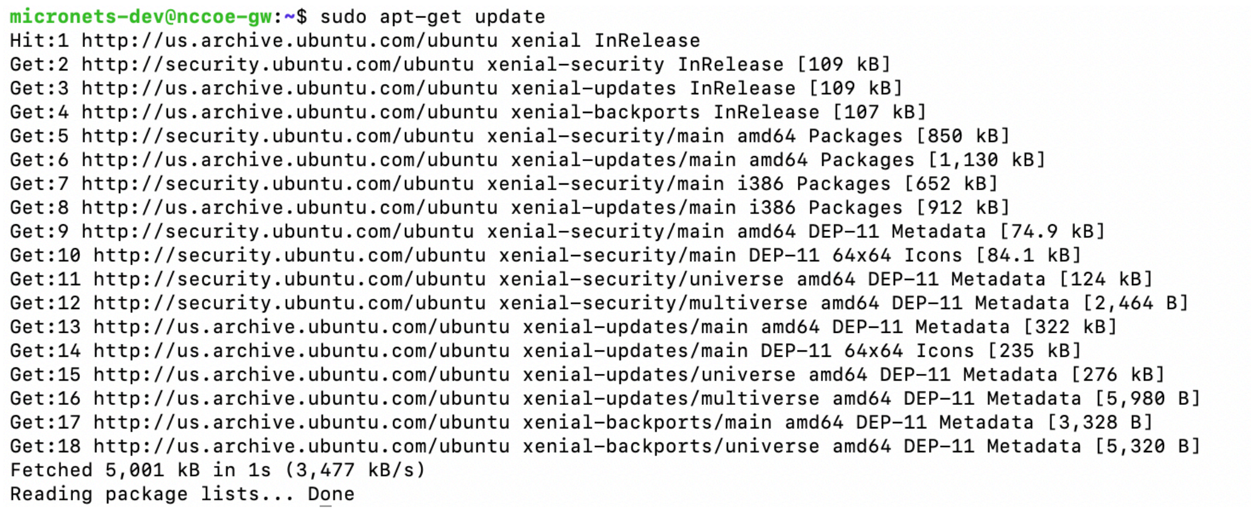
1. If Micronets is already installed and running, you should stop the services first by executing the following commands:

   ```
   sudo systemctl stop micronets-gw.service


   sudo systemctl stop micronets-hostapd.service
   ```

2. Update your local package index by entering the following command:

   ```
   sudo apt-get update
   ```

   You should see the following output from this command:

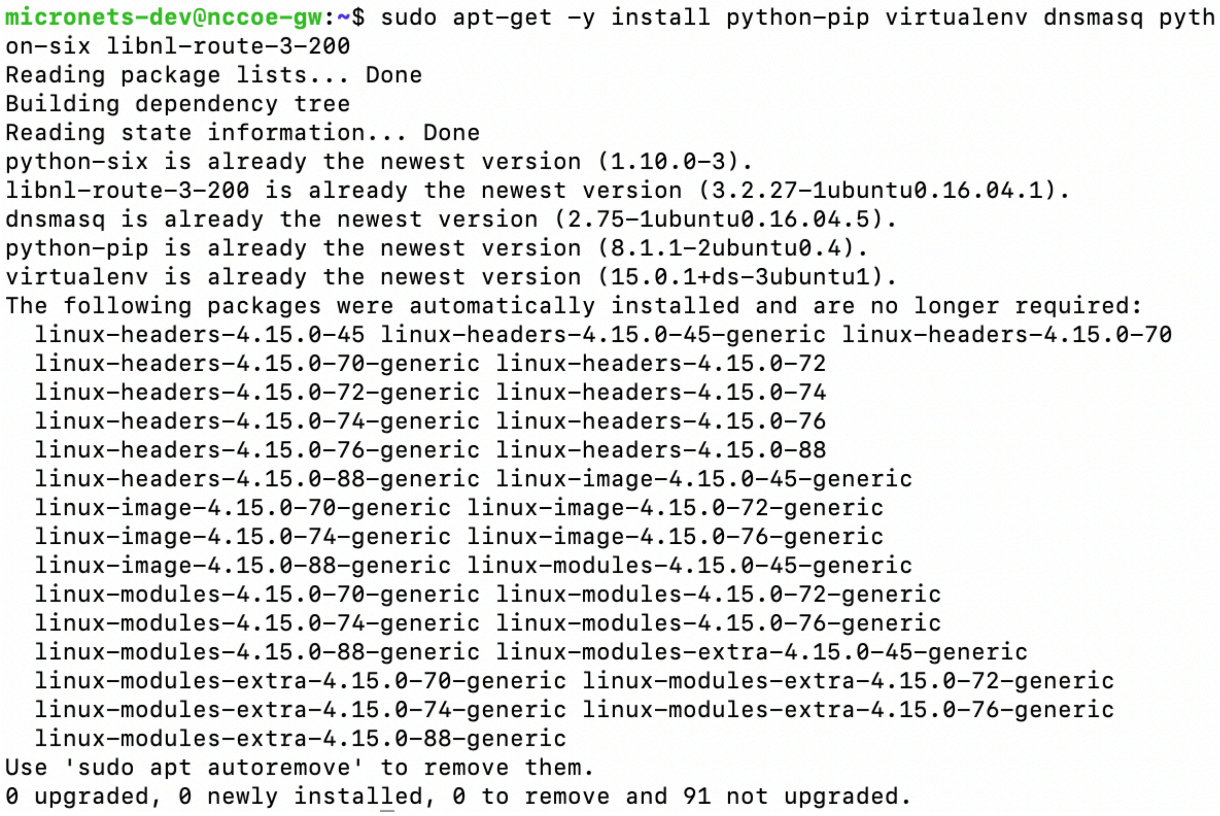


```
micronets-dev@nccoe-gw:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:5 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [850 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [1,130 kB]
Get:7 http://security.ubuntu.com/ubuntu xenial-security/main i386 Packages [652 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu xenial-updates/main i386 Packages [912 kB]
Get:9 http://security.ubuntu.com/ubuntu xenial-security/main amd64 DEP-11 Metadata [74.9 kB]
Get:10 http://security.ubuntu.com/ubuntu xenial-security/main DEP-11 64x64 Icons [84.1 kB]
Get:11 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 DEP-11 Metadata [124 kB]
Get:12 http://security.ubuntu.com/ubuntu xenial-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:13 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 DEP-11 Metadata [322 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu xenial-updates/main DEP-11 64x64 Icons [235 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 DEP-11 Metadata [276 kB]
Get:16 http://us.archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 DEP-11 Metadata [5,980 B]
Get:17 http://us.archive.ubuntu.com/ubuntu xenial-backports/main amd64 DEP-11 Metadata [3,328 B]
Get:18 http://us.archive.ubuntu.com/ubuntu xenial-backports/universe amd64 DEP-11 Metadata [5,320 B]
Fetched 5,001 kB in 1s (3,477 kB/s)
Reading package lists... Done
```

3. Install the **python-pip, virtualenv, dnsmasq, python-six,** and **libnl-route-3-200** packages by executing the following command:

   ```
   sudo apt-get -y install python-pip virtualenv dnsmasq python-six libnl-route-3-200
   ```

   If the packages are not already installed, you should see the following output from this command:

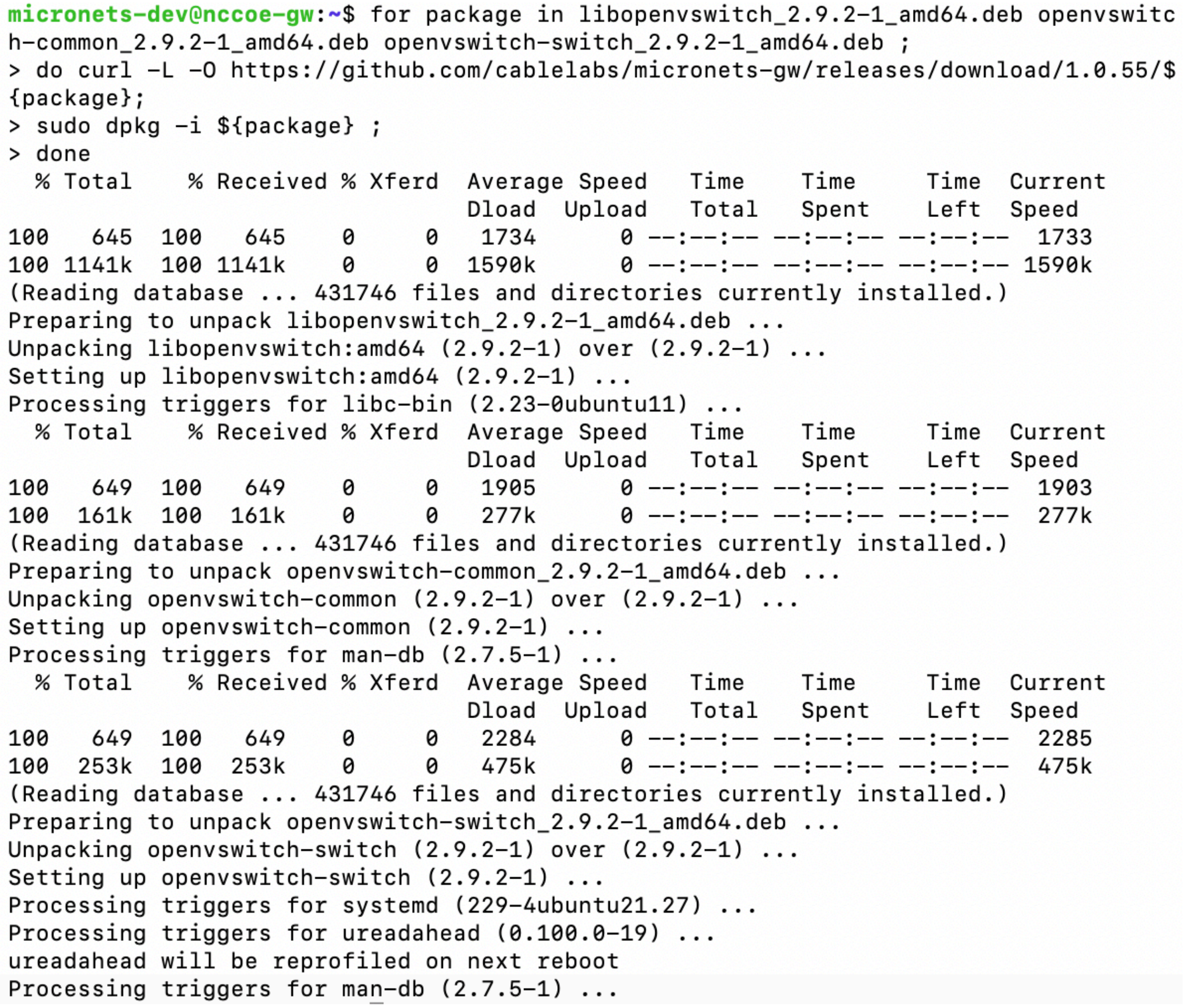


```
micronets-dev@nccoe-gw:~$ sudo apt-get -y install python-pip virtualenv dnsmasq pyth
on-six libnl-route-3-200
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-six is already the newest version (1.10.0-3).
libnl-route-3-200 is already the newest version (3.2.27-1ubuntu0.16.04.1).
dnsmasq is already the newest version (2.75-1ubuntu0.16.04.5).
python-pip is already the newest version (8.1.1-2ubuntu0.4).
virtualenv is already the newest version (15.0.1+ds-3ubuntu1).
The following packages were automatically installed and are no longer required:
  linux-headers-4.15.0-45 linux-headers-4.15.0-45-generic linux-headers-4.15.0-70
  linux-headers-4.15.0-70-generic linux-headers-4.15.0-72
  linux-headers-4.15.0-72-generic linux-headers-4.15.0-74
  linux-headers-4.15.0-74-generic linux-headers-4.15.0-76
  linux-headers-4.15.0-76-generic linux-headers-4.15.0-88
  linux-headers-4.15.0-88-generic linux-image-4.15.0-45-generic
  linux-image-4.15.0-70-generic linux-image-4.15.0-72-generic
  linux-image-4.15.0-74-generic linux-image-4.15.0-76-generic
  linux-image-4.15.0-88-generic linux-modules-4.15.0-45-generic
  linux-modules-4.15.0-70-generic linux-modules-4.15.0-72-generic
  linux-modules-4.15.0-74-generic linux-modules-4.15.0-76-generic
  linux-modules-4.15.0-88-generic linux-modules-extra-4.15.0-45-generic
  linux-modules-extra-4.15.0-70-generic linux-modules-extra-4.15.0-72-generic
  linux-modules-extra-4.15.0-74-generic linux-modules-extra-4.15.0-76-generic
  linux-modules-extra-4.15.0-88-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 91 not upgraded.
```

4. Install openvswitch version 2.9.2 and its dependencies from the CableLabs micronets-gw github repository by executing the following for loop:

```
for package in libopenvswitch_2.9.2-1_amd64.deb \
                openvswitch-common_2.9.2-1_amd64.deb \
                openvswitch-switch_2.9.2-1_amd64.deb ;
do curl -L -O https://github.com/cablelabs/micronets-gw/releases/down-
load/1.0.55/${package};
sudo dpkg -i ${package};
done
```

You should see the following output from this command:

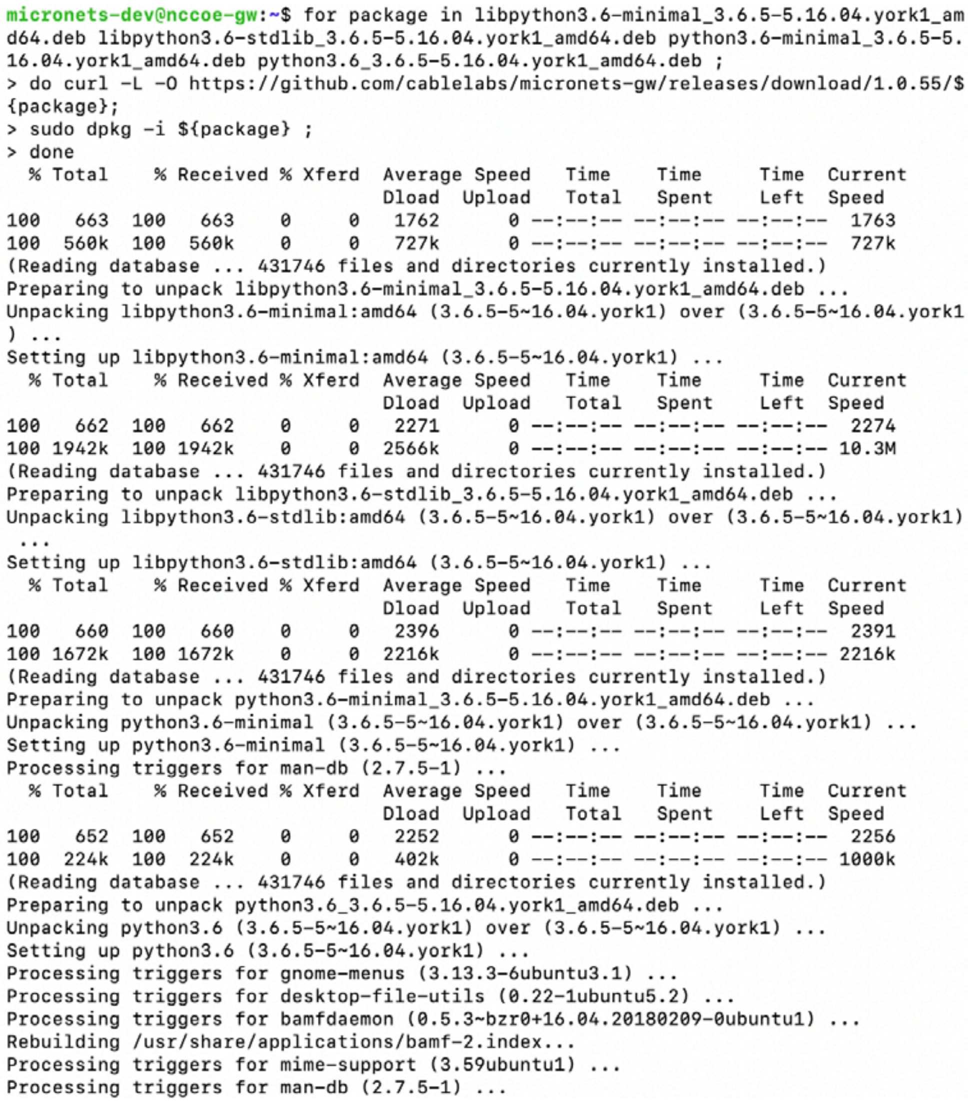


```
micronets-dev@nccoe-gw:~$ for package in libopenvswitch_2.9.2-1_amd64.deb openvswitc
h-common_2.9.2-1_amd64.deb openvswitch-switch_2.9.2-1_amd64.deb ;
> do curl -L -O https://github.com/cablelabs/micronets-gw/releases/download/1.0.55/$
{package};
> sudo dpkg -i ${package} ;
> done
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   645  100   645    0     0   1734      0 --:--:-- --:--:-- --:--:--  1733
100 1141k  100 1141k    0     0  1590k      0 --:--:-- --:--:-- --:--:-- 1590k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack libopenvswitch_2.9.2-1_amd64.deb ...
Unpacking libopenvswitch:amd64 (2.9.2-1) over (2.9.2-1) ...
Setting up libopenvswitch:amd64 (2.9.2-1) ...
Processing triggers for libc-bin (2.23-0ubuntu11) ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   649  100   649    0     0   1905      0 --:--:-- --:--:-- --:--:--  1903
100  161k  100  161k    0     0   277k      0 --:--:-- --:--:-- --:--:--  277k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack openvswitch-common_2.9.2-1_amd64.deb ...
Unpacking openvswitch-common (2.9.2-1) over (2.9.2-1) ...
Setting up openvswitch-common (2.9.2-1) ...
Processing triggers for man-db (2.7.5-1) ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   649  100   649    0     0   2284      0 --:--:-- --:--:-- --:--:--  2285
100  253k  100  253k    0     0   475k      0 --:--:-- --:--:-- --:--:--  475k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack openvswitch-switch_2.9.2-1_amd64.deb ...
Unpacking openvswitch-switch (2.9.2-1) over (2.9.2-1) ...
Setting up openvswitch-switch (2.9.2-1) ...
Processing triggers for systemd (229-4ubuntu21.27) ...
Processing triggers for ureadahead (0.100.0-19) ...
ureadahead will be reprofiled on next reboot
Processing triggers for man-db (2.7.5-1) ...
```

5. Install Python version 3.6 and its dependencies from the CableLabs micronets-gw github repository by executing the following for loop:

```
for package in libpython3.6-minimal_3.6.5-5.16.04.york1_amd64.deb \
               libpython3.6-stdlib_3.6.5-5.16.04.york1_amd64.deb \
               python3.6-minimal_3.6.5-5.16.04.york1_amd64.deb \
               python3.6_3.6.5-5.16.04.york1_amd64.deb ;
do curl -L -O https://github.com/cablelabs/micronets-gw/releases/down-
load/1.0.55/${package};
```

You should see the following output from this command:



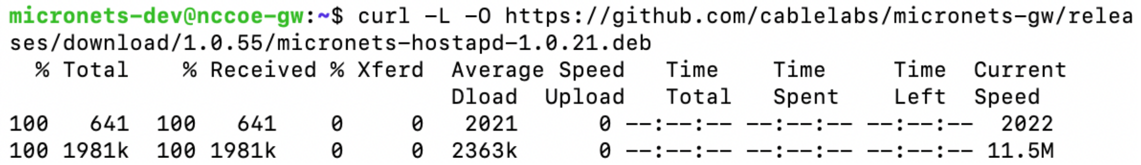
```
micronets-dev@nccoe-gw:~$ for package in libpython3.6-minimal_3.6.5-5.16.04.york1_am
d64.deb libpython3.6-stdlib_3.6.5-5.16.04.york1_amd64.deb python3.6-minimal_3.6.5-5.
16.04.york1_amd64.deb python3.6_3.6.5-5.16.04.york1_amd64.deb ;
> do curl -L -O https://github.com/cablelabs/micronets-gw/releases/download/1.0.55/$
{package};
> sudo dpkg -i ${package} ;
> done
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   663  100   663    0     0   1762      0 --:--:-- --:--:-- --:--:--  1763
100  560k  100  560k    0     0   727k      0 --:--:-- --:--:-- --:--:--  727k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack libpython3.6-minimal_3.6.5-5.16.04.york1_amd64.deb ...
Unpacking libpython3.6-minimal:amd64 (3.6.5-5~16.04.york1) over (3.6.5-5~16.04.york1
) ...
Setting up libpython3.6-minimal:amd64 (3.6.5-5~16.04.york1) ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   662  100   662    0     0   2271      0 --:--:-- --:--:-- --:--:--  2274
100 1942k  100 1942k    0     0  2566k      0 --:--:-- --:--:-- --:--:-- 10.3M
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack libpython3.6-stdlib_3.6.5-5.16.04.york1_amd64.deb ...
Unpacking libpython3.6-stdlib:amd64 (3.6.5-5~16.04.york1) over (3.6.5-5~16.04.york1)
 ...
Setting up libpython3.6-stdlib:amd64 (3.6.5-5~16.04.york1) ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   660  100   660    0     0   2396      0 --:--:-- --:--:-- --:--:--  2391
100 1672k  100 1672k    0     0  2216k      0 --:--:-- --:--:-- --:--:-- 2216k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack python3.6-minimal_3.6.5-5.16.04.york1_amd64.deb ...
Unpacking python3.6-minimal (3.6.5-5~16.04.york1) over (3.6.5-5~16.04.york1) ...
Setting up python3.6-minimal (3.6.5-5~16.04.york1) ...
Processing triggers for man-db (2.7.5-1) ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   652  100   652    0     0   2252      0 --:--:-- --:--:-- --:--:--  2256
100  224k  100  224k    0     0   402k      0 --:--:-- --:--:-- --:--:-- 1000k
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack python3.6_3.6.5-5.16.04.york1_amd64.deb ...
Unpacking python3.6 (3.6.5-5~16.04.york1) over (3.6.5-5~16.04.york1) ...
Setting up python3.6 (3.6.5-5~16.04.york1) ...
Processing triggers for gnome-menus (3.13.3-6ubuntu3.1) ...
Processing triggers for desktop-file-utils (0.22-1ubuntu5.2) ...
Processing triggers for bamfdaemon (0.5.3~bzr0+16.04.20180209-0ubuntu1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for mime-support (3.59ubuntu1) ...
Processing triggers for man-db (2.7.5-1) ...
```

4.1.4.3.2 Install Micronets Packages

1. Enter the following command to download the Micronets hostapd package:

```
curl -L -O https://github.com/cablelabs/micronets-gw/releases/down-
load/1.0.55/micronets-hostapd-1.0.21.deb
```

You should see output similar to the following:

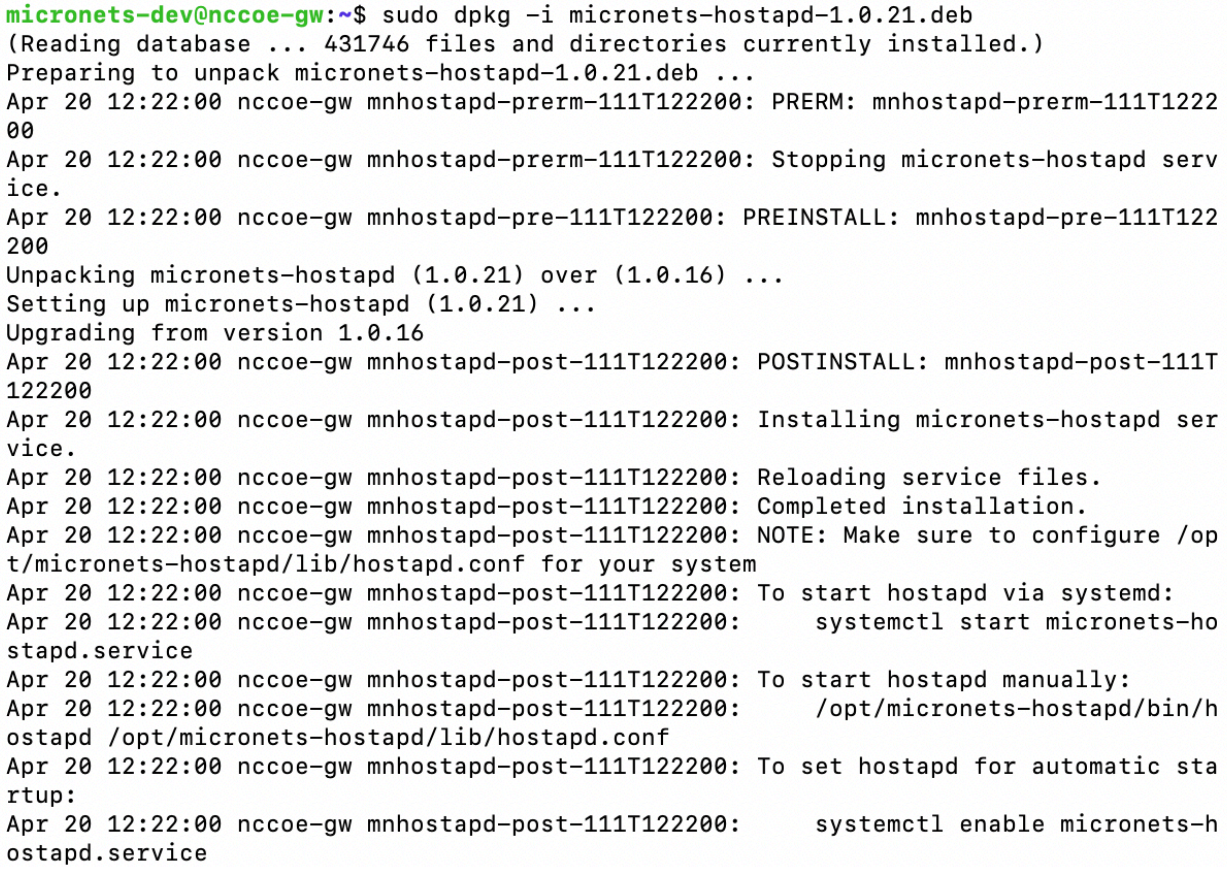


```
micronets-dev@nccoe-gw:~$ curl -L -O https://github.com/cablelabs/micronets-gw/relea
ses/download/1.0.55/micronets-hostapd-1.0.21.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   641  100   641    0     0   2021      0 --:--:-- --:--:-- --:--:--  2022
100 1981k  100 1981k    0     0  2363k      0 --:--:-- --:--:-- --:--:-- 11.5M
```

2. Enter the following command to de-package the Micronets hostapd package:

```
sudo dpkg -i micronets-hostapd-1.0.21.deb
```

You should see output similar to the following:

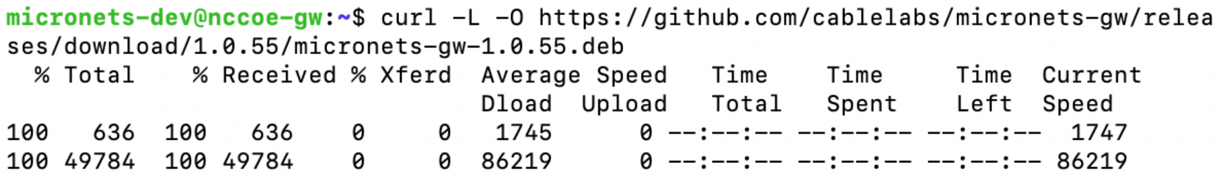


```
micronets-dev@nccoe-gw:~$ sudo dpkg -i micronets-hostapd-1.0.21.deb
(Reading database ... 431746 files and directories currently installed.)
Preparing to unpack micronets-hostapd-1.0.21.deb ...
Apr 20 12:22:00 nccoe-gw mnhostapd-prerm-111T122200: PRERM: mnhostapd-prerm-111T1222
00
Apr 20 12:22:00 nccoe-gw mnhostapd-prerm-111T122200: Stopping micronets-hostapd serv
ice.
Apr 20 12:22:00 nccoe-gw mnhostapd-pre-111T122200: PREINSTALL: mnhostapd-pre-111T122
200
Unpacking micronets-hostapd (1.0.21) over (1.0.16) ...
Setting up micronets-hostapd (1.0.21) ...
Upgrading from version 1.0.16
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: POSTINSTALL: mnhostapd-post-111T
122200
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: Installing micronets-hostapd ser
vice.
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: Reloading service files.
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: Completed installation.
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: NOTE: Make sure to configure /op
t/micronets-hostapd/lib/hostapd.conf for your system
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: To start hostapd via systemd:
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200:     systemctl start micronets-ho
stapd.service
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: To start hostapd manually:
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200:     /opt/micronets-hostapd/bin/h
ostapd /opt/micronets-hostapd/lib/hostapd.conf
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200: To set hostapd for automatic sta
rtup:
Apr 20 12:22:00 nccoe-gw mnhostapd-post-111T122200:     systemctl enable micronets-h
ostapd.service
```

3. Enter the following command to download the Micronets Gateway package:

```
curl -L -O https://github.com/cablelabs/micronets-gw/releases/down-
load/1.0.55/micronets-gw-1.0.55.deb
```

You should see output similar to the following:

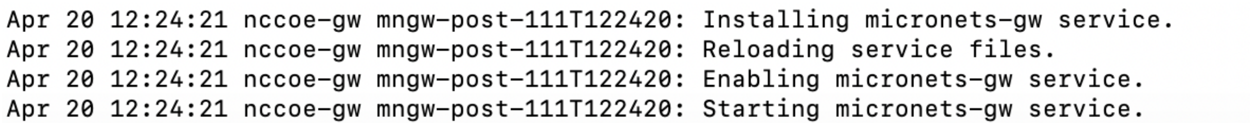


```
micronets-dev@nccoe-gw:~$ curl -L -O https://github.com/cablelabs/micronets-gw/relea
ses/download/1.0.55/micronets-gw-1.0.55.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   636  100   636    0     0   1745      0 --:--:-- --:--:-- --:--:--  1747
100 49784  100 49784    0     0  86219      0 --:--:-- --:--:-- --:--:-- 86219
```

4. Enter the following command to install the Micronets hostapd package:

   ```
   sudo dpkg -i micronets-gw-1.0.55.deb
   ```

   After a bit of a delay, you should see output similar to the following:



```
Apr 20 12:24:21 nccoe-gw mngw-post-111T122420: Installing micronets-gw service.
Apr 20 12:24:21 nccoe-gw mngw-post-111T122420: Reloading service files.
Apr 20 12:24:21 nccoe-gw mngw-post-111T122420: Enabling micronets-gw service.
Apr 20 12:24:21 nccoe-gw mngw-post-111T122420: Starting micronets-gw service.
```

5. Enable autostart for the Micronets hostapd service by entering the following command:

   ```
   sudo systemctl enable micronets-hostapd.service
   ```

6. Enable autostart for the Micronets Gateway Service by entering the following command:

   ```
   sudo systemctl enable micronets-gw.service
   ```

7. Start the Micronets hostapd service by entering the following command:

   ```
   sudo systemctl start micronets-hostapd.service
   ```

8. Start the Micronets Gateway Service by entering the following command:

   ```
    sudo systemctl start micronets-gw.service
   ```

9. Verify that the gateway service started successfully by running the following command:

   ```
   sudo systemctl status micronets-gw.service
   ```

10. Verify that the Micronets hostapd service started successfully by running the following command:

    ```
    sudo systemctl status micronets-hostapd.service
    ```

CableLabs documentation notes that installing the micronets-gw package should produce the following results:

- installation of the Micronets Gateway Service in the /opt/micronets-gw directory
- installation of the ifup/down and dnsmasq extension scripts for configuration of openvswitch and the micronets-gw service via /etc/network/interfaces
- installation of a sample/etc/network/interfaces file in /opt/micronets-gw/doc/interfaces.sample
- installation and start of the micronets-gw-service systemd service

4.1.5 IoT Devices

This section provides configuration details for the Linux-based IoT development kits used in the build, which can be onboarded via DPP. It also provides information regarding a basic IoT application used to test the MUD process.

4.1.5.1 IoT Devices Overview

Build 3, like the other builds in this project, leverages the Raspberry Pi devkit with capabilities developed to make these devices both MUD- and DPP-capable. The Raspberry Pi runs the Raspbian 9 OS and is provisioned with one bootstrapping public/private key pair during device setup. The Micronets Proto-Pi software developed by CableLabs in combination with the added hardware outlined in the configuration section adds DPP capability to these devices. There are two onboarding mechanisms called *modes* supported by the Micronets Proto-Pi software: DPP mode and clinic mode. The clinic mode provides an onboarding mechanism via automated installation of Wi-Fi security certificates, and the DPP mode provides QR code–based device onboarding. For this implementation, we only describe setting up and leveraging the Micronets Proto-Pi software in DPP mode. If you would like to leverage the clinic mode of this software, follow the documentation provided by CableLabs: https://github.com/cablelabs/micronets-pi3/blob/nccoe-build-3/README.md#Installation.

4.1.5.2 Configuration Overview

The following subsections document the software and network configurations for the Micronets Proto-Pi device.

4.1.5.2.1 Network Configuration

The following network configurations are required to install, configure, and operate the Micronets Proto-Pi device:

- wired network connection to a separate access point that provides both initial internet access to self-register the device and remote management access to the device during setup

4.1.5.2.2 Software Configuration

The following software is required to install, configure, and operate the Micronets Proto-Pi device:

- tool for flashing images to Secure Digital (SD) card (This implementation leveraged balenaEtcher: https://www.balena.io/etcher/.)
- latest Raspbian image from:
    - CableLabs at the following link (This image has Secure Shell (SSH) and Visual (vi) preinstalled): https://www.dropbox.com/s/37ygauo02ltxirf/raspbian-buster-ssh-updates.zip?dl=0
    - Or you can download the latest Buster distribution and install packages yourself from the following link: https://www.raspberrypi.org/downloads/raspbian/

4.1.5.2.3 Hardware Configuration

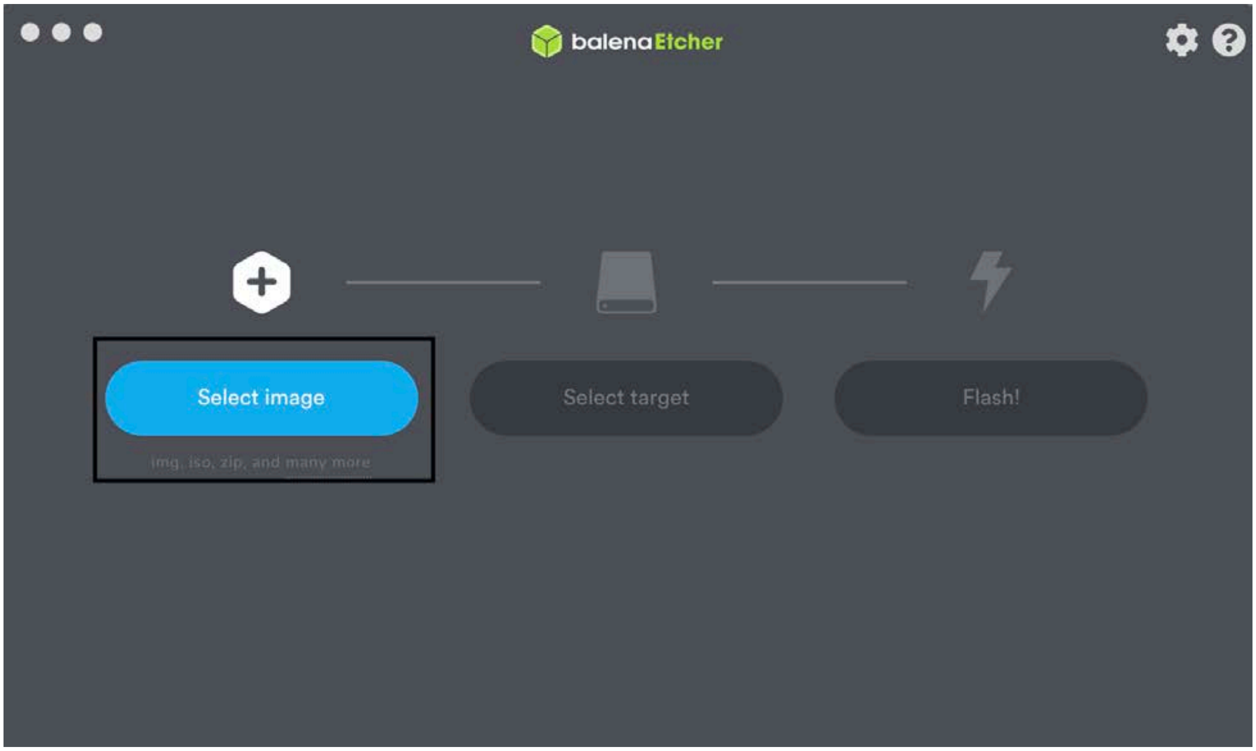
The following hardware is required to install, configure, and operate the Micronets Proto-Pi device:

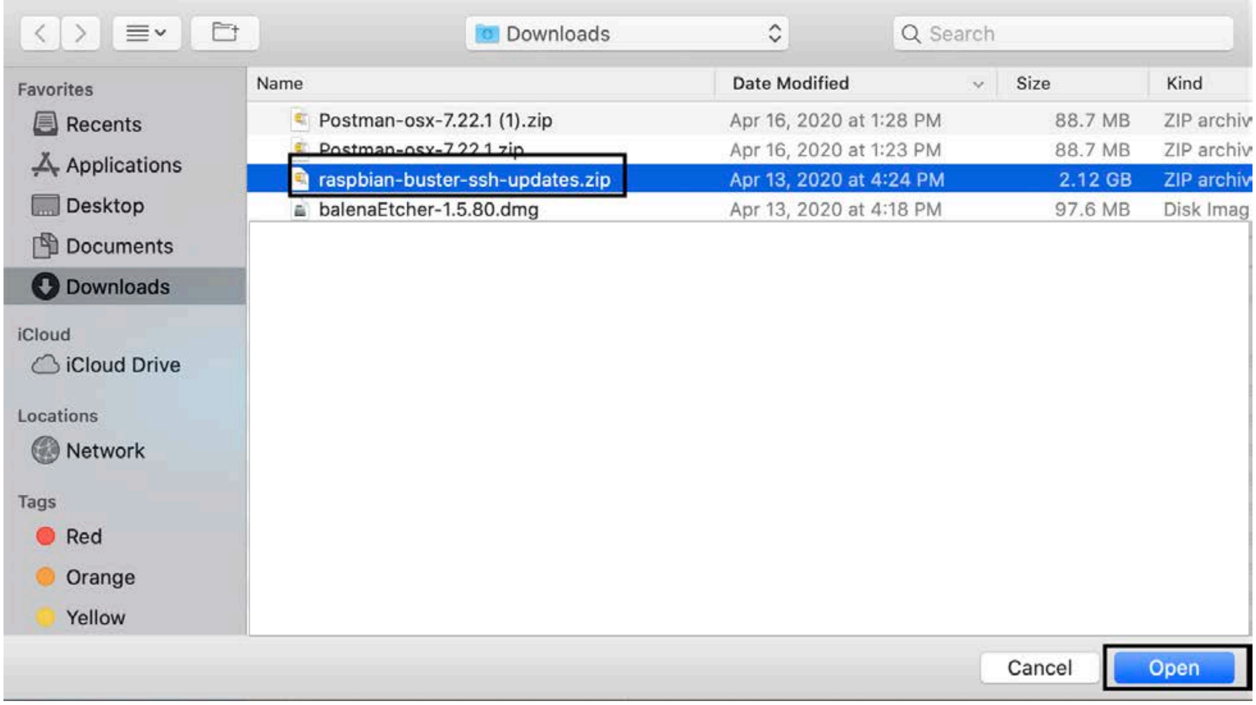
- Raspberry Pi (version 3B+)
- SD card
- Alfa adapter
- Ethernet cable

4.1.5.3 Setup

4.1.5.3.1 Install Dependencies

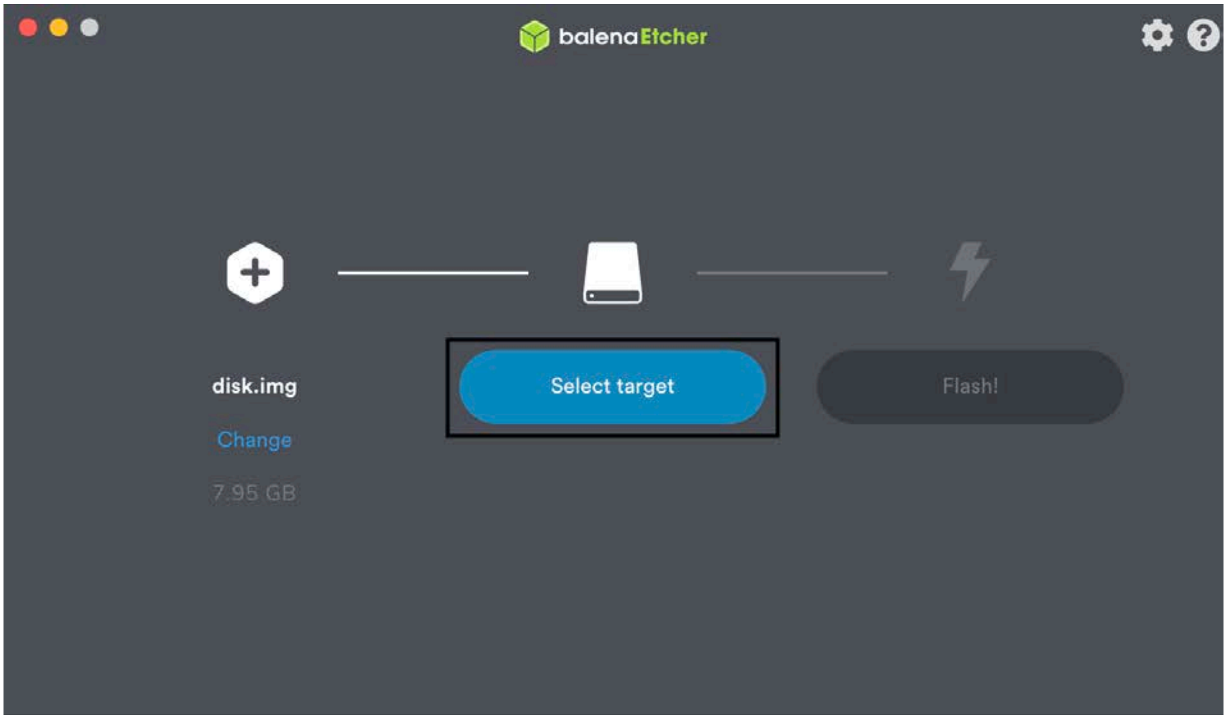
1. Connect the SD card to your computer.
2. Open balenaEtcher (or whatever tool you have downloaded for flashing SD cards).
3. Click **Select image,** and select the Raspbian image you downloaded:



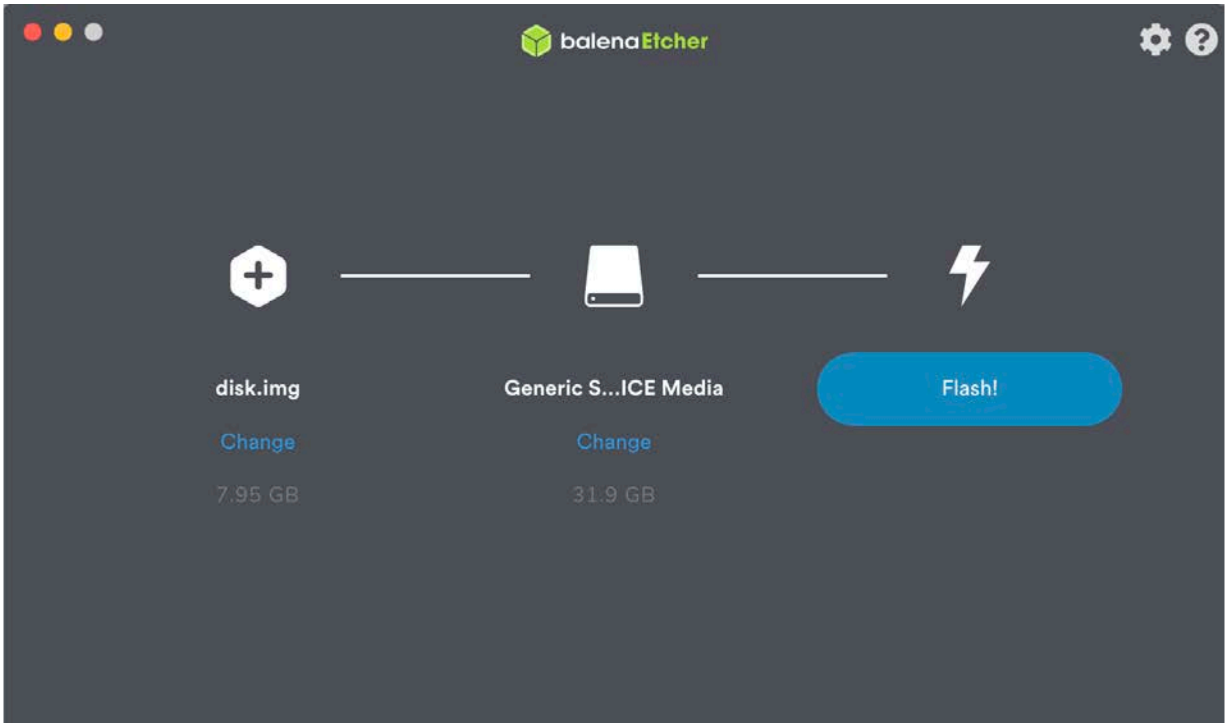


Name	Date Modified	Size	Kind
Postman-osx-7.22.1 (1).zip	Apr 16, 2020 at 1:28 PM	88.7 MB	ZIP archiv
Postman-osx-7.22.1.zip	Apr 16, 2020 at 1:23 PM	88.7 MB	ZIP archiv
raspbian-buster-ssh-updates.zip	Apr 13, 2020 at 4:24 PM	2.12 GB	ZIP archiv
balenaEtcher-1.5.80.dmg	Apr 13, 2020 at 4:18 PM	97.6 MB	Disk Imag

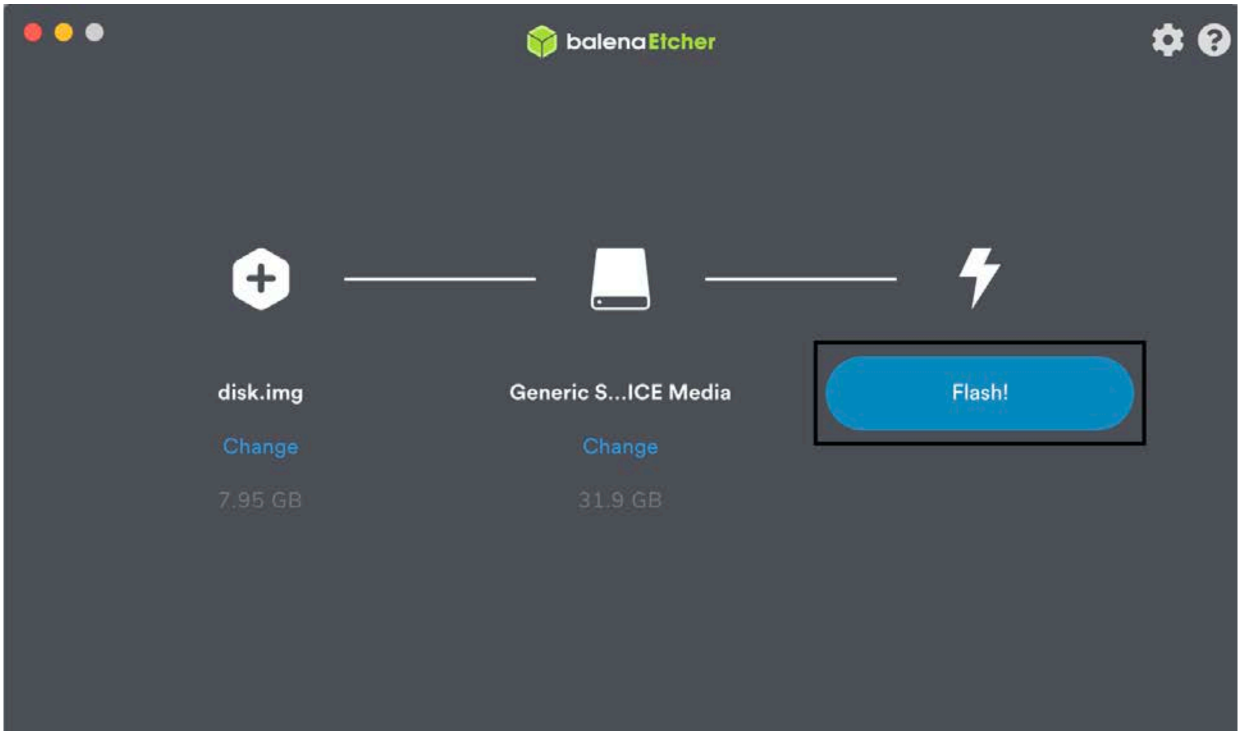
4. Click **Select target,** and select the SD card you connected to the computer (the software may automatically recognize the target):



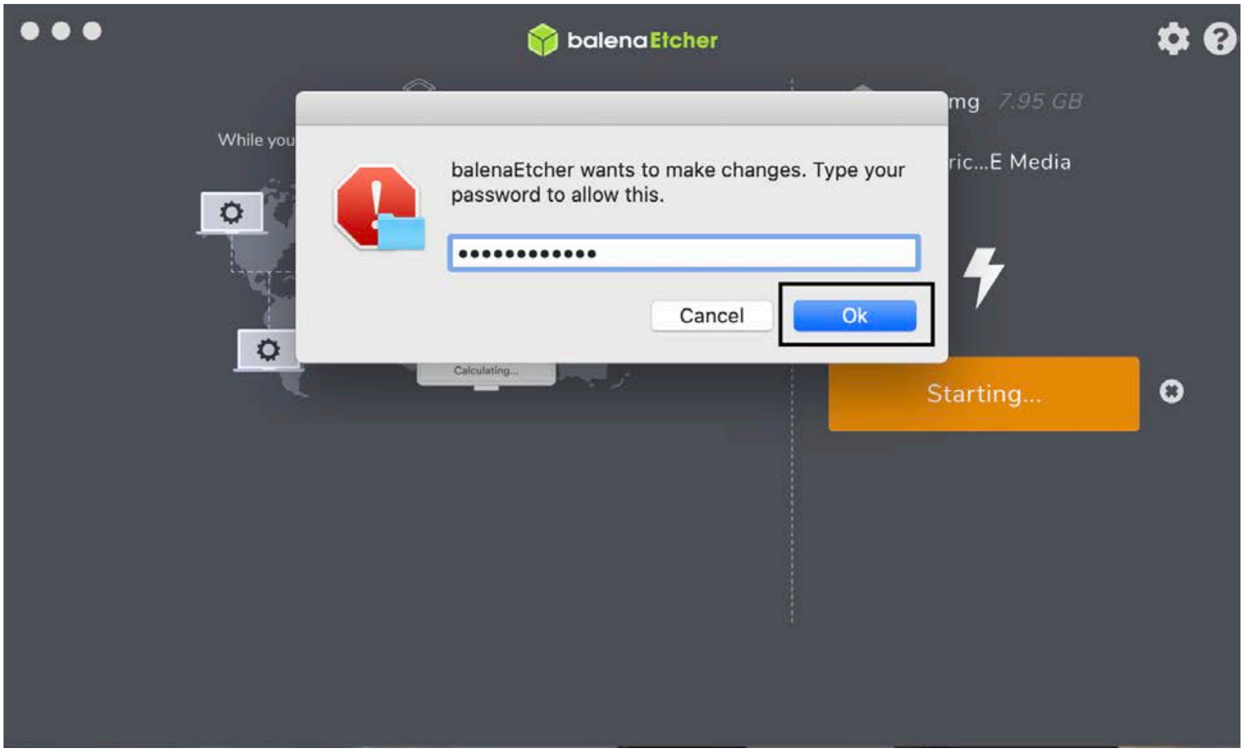
You should see something similar to the following:



2055 5. Click **Flash!** to start the flashing process:



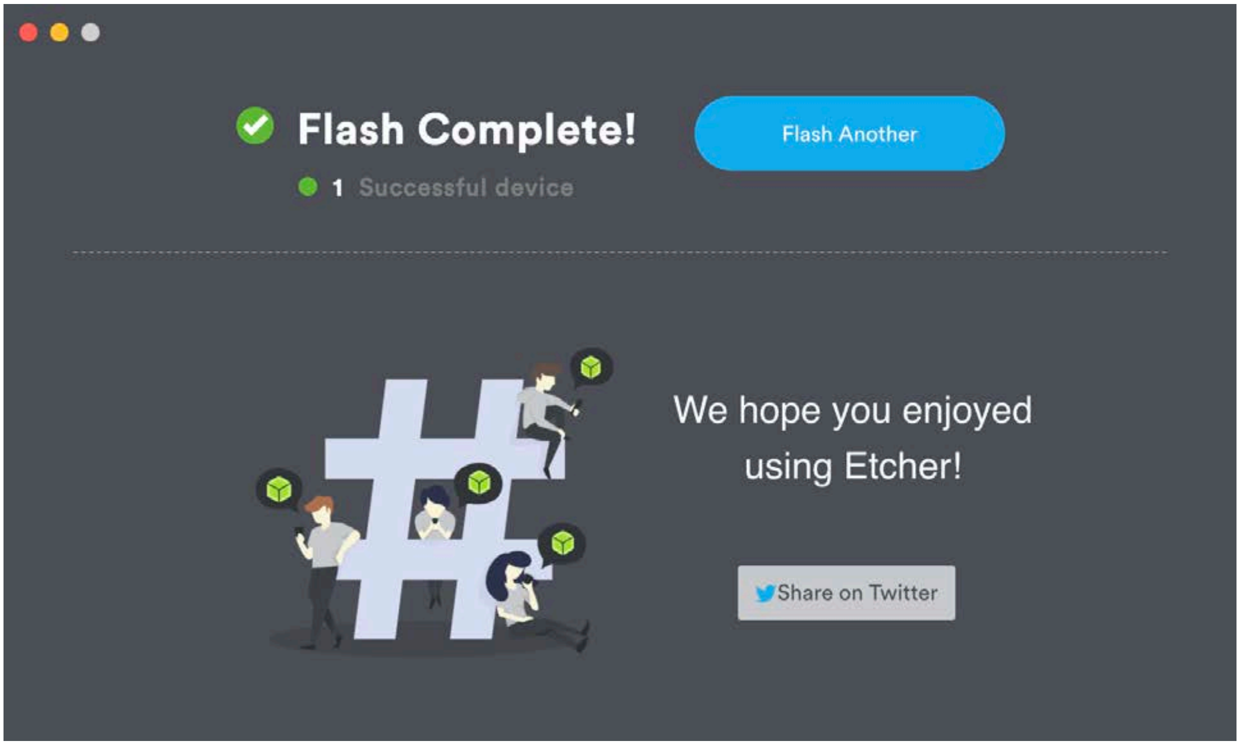
2056 You may be prompted to enter your password, as seen below:



balenaEtcher wants to make changes. Type your password to allow this.

•••••••••••••

2057 When the flashing has completed, you should see output similar to the following:



Flash Complete!

1 Successful device

We hope you enjoyed using Etcher!

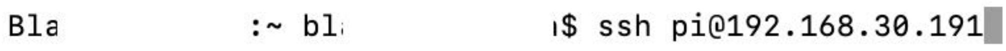
4.1.5.3.2 Install Micronets Proto-Pi

1. Insert the SD card to the Raspberry Pi, and connect power using a micro–Universal Serial Bus (USB) cable.
2. Connect to the Raspberry Pi from a remote machine by using SSH:

   Note: You will need to figure out the Ethernet IP address of the Raspberry Pi, which can be done by looking at the DHCP assignments on the gateway to which you connected the Raspberry Pi.

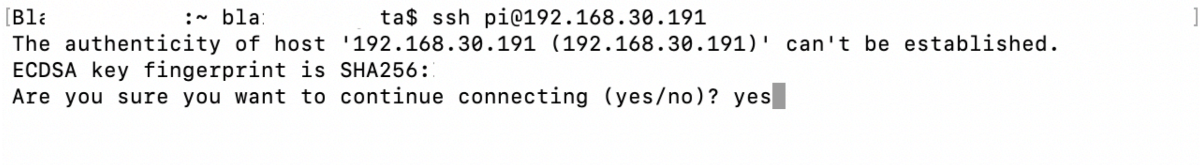
   a. Enter the following command once you have identified the device’s IP address:

      ```
      ssh pi@[ipaddress]
      ```

      ```
      Bla          :~ bl          ı$ ssh pi@192.168.30.191
      ```

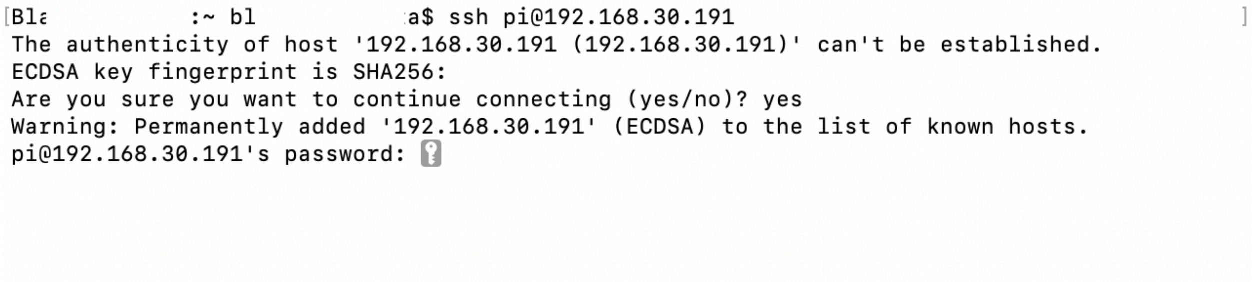
   b. You will be prompted to continue connecting as this is the first time connecting to the device:



```
[Bla          :~ bla         ta$ ssh pi@192.168.30.191                              ]
The authenticity of host '192.168.30.191 (192.168.30.191)' can't be established.
ECDSA key fingerprint is SHA256:
Are you sure you want to continue connecting (yes/no)? yes
```

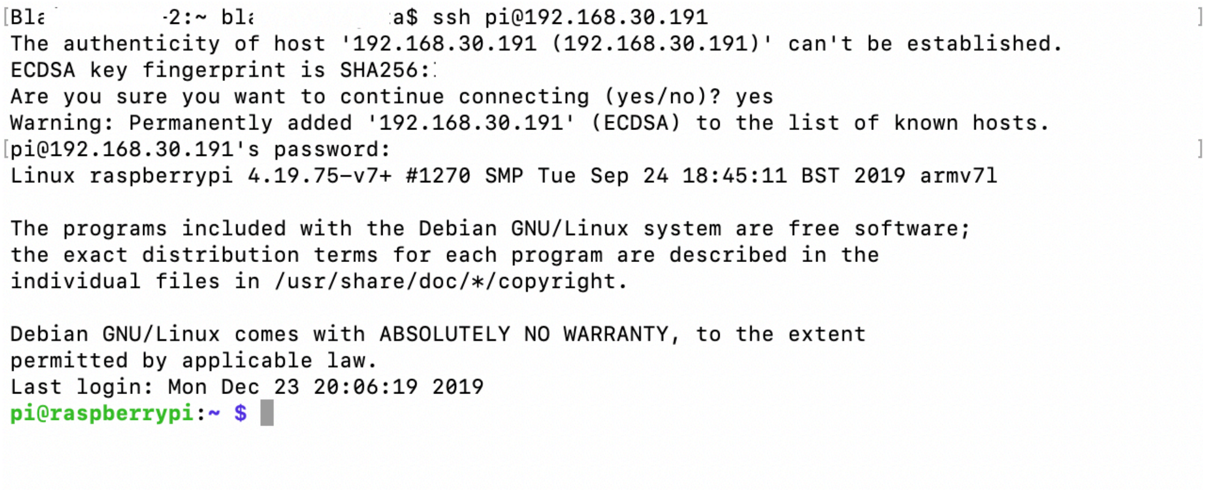
c. Enter the password for the Raspberry Pi:

Note: The password is "micronets" if you are leveraging the CableLabs Raspberry Pi image:



```
[Bla          :~ bl           a$ ssh pi@192.168.30.191                                  ]
The authenticity of host '192.168.30.191 (192.168.30.191)' can't be established.
ECDSA key fingerprint is SHA256:
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.30.191' (ECDSA) to the list of known hosts.
pi@192.168.30.191's password: 
```

d. You will now have access to a terminal on the Raspberry Pi:



```
[Bla         -2:~ bla           a$ ssh pi@192.168.30.191                            ]
The authenticity of host '192.168.30.191 (192.168.30.191)' can't be established.
ECDSA key fingerprint is SHA256:
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.30.191' (ECDSA) to the list of known hosts.
[pi@192.168.30.191's password:                                                      ]
Linux raspberrypi 4.19.75-v7+ #1270 SMP Tue Sep 24 18:45:11 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 23 20:06:19 2019
pi@raspberrypi:~ $ 
```

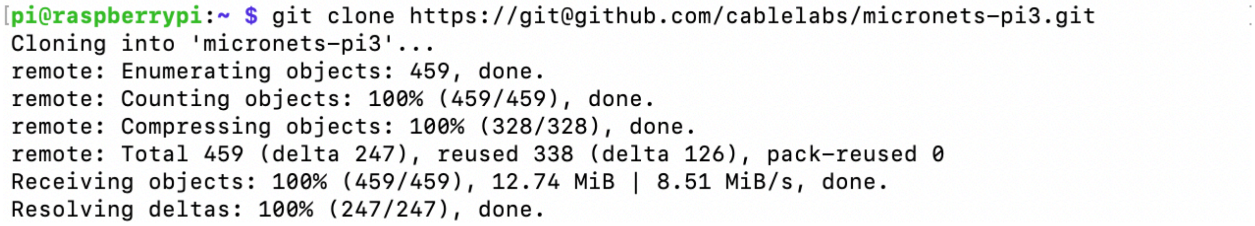
3. Ensure that you are in the home directory by entering the following command:

```
cd ~
```

4. Download the Micronets Proto-Pi software from GitHub by entering the following command:

```
git clone https://git@github.com/cablelabs/micronets-pi3.git
```

You should see output similar to the following:



```
[pi@raspberrypi:~ $ git clone https://git@github.com/cablelabs/micronets-pi3.git
Cloning into 'micronets-pi3'...
remote: Enumerating objects: 459, done.
remote: Counting objects: 100% (459/459), done.
remote: Compressing objects: 100% (328/328), done.
remote: Total 459 (delta 247), reused 338 (delta 126), pack-reused 0
Receiving objects: 100% (459/459), 12.74 MiB | 8.51 MiB/s, done.
Resolving deltas: 100% (247/247), done.
```

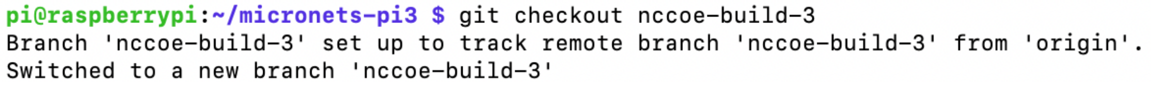
5. Change into the micronets-pi3 directory by entering the following command:

```
cd micronets-pi3/
```

6. Check out the nccoe-build-3 branch by entering the following branch:

```
git checkout nccoe-build-3
```

You should see output similar to the following:



```
pi@raspberrypi:~/micronets-pi3 $ git checkout nccoe-build-3
Branch 'nccoe-build-3' set up to track remote branch 'nccoe-build-3' from 'origin'.
Switched to a new branch 'nccoe-build-3'
```

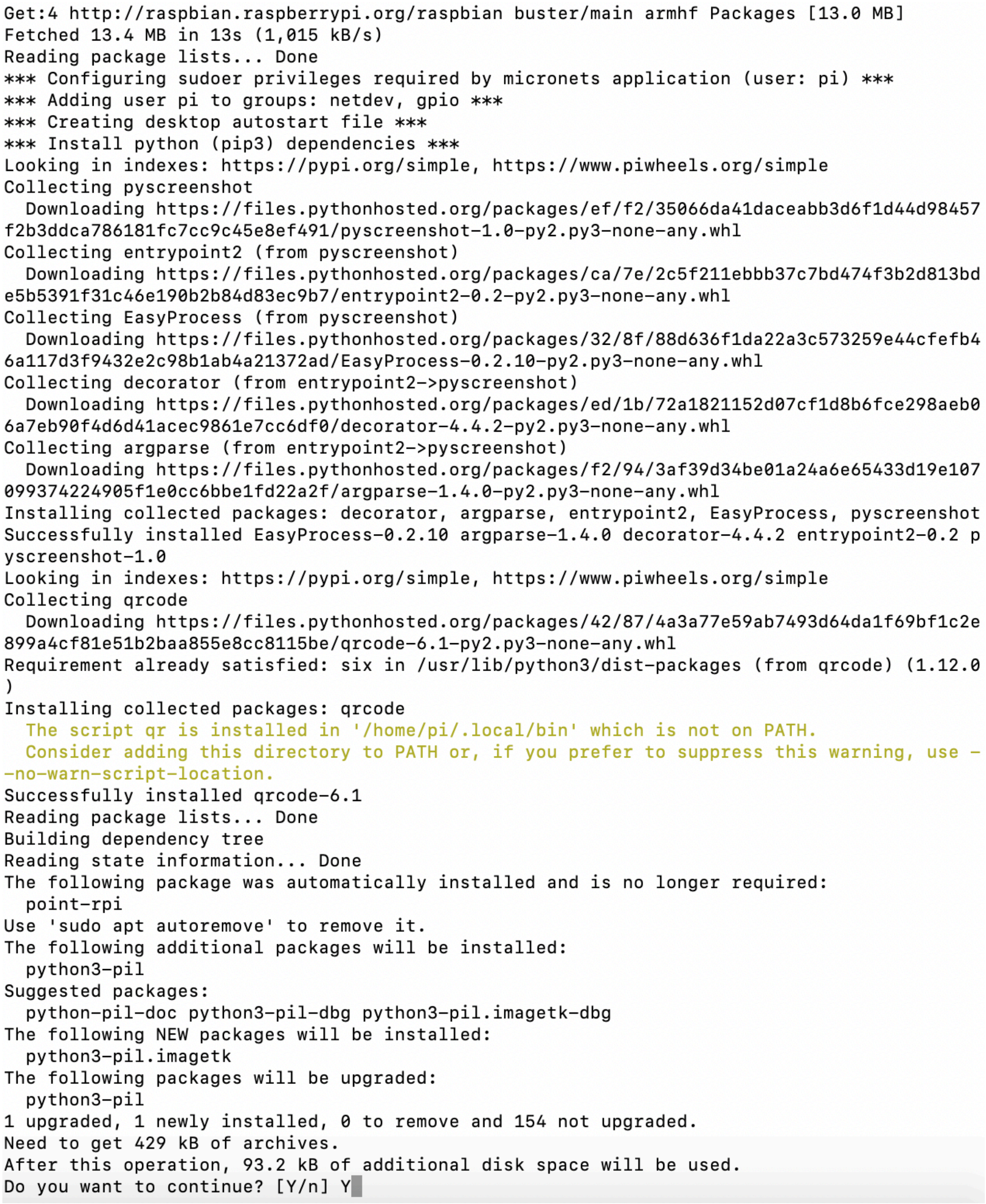
7. Change into the deploy directory by entering the following command:

```
cd deploy/
```

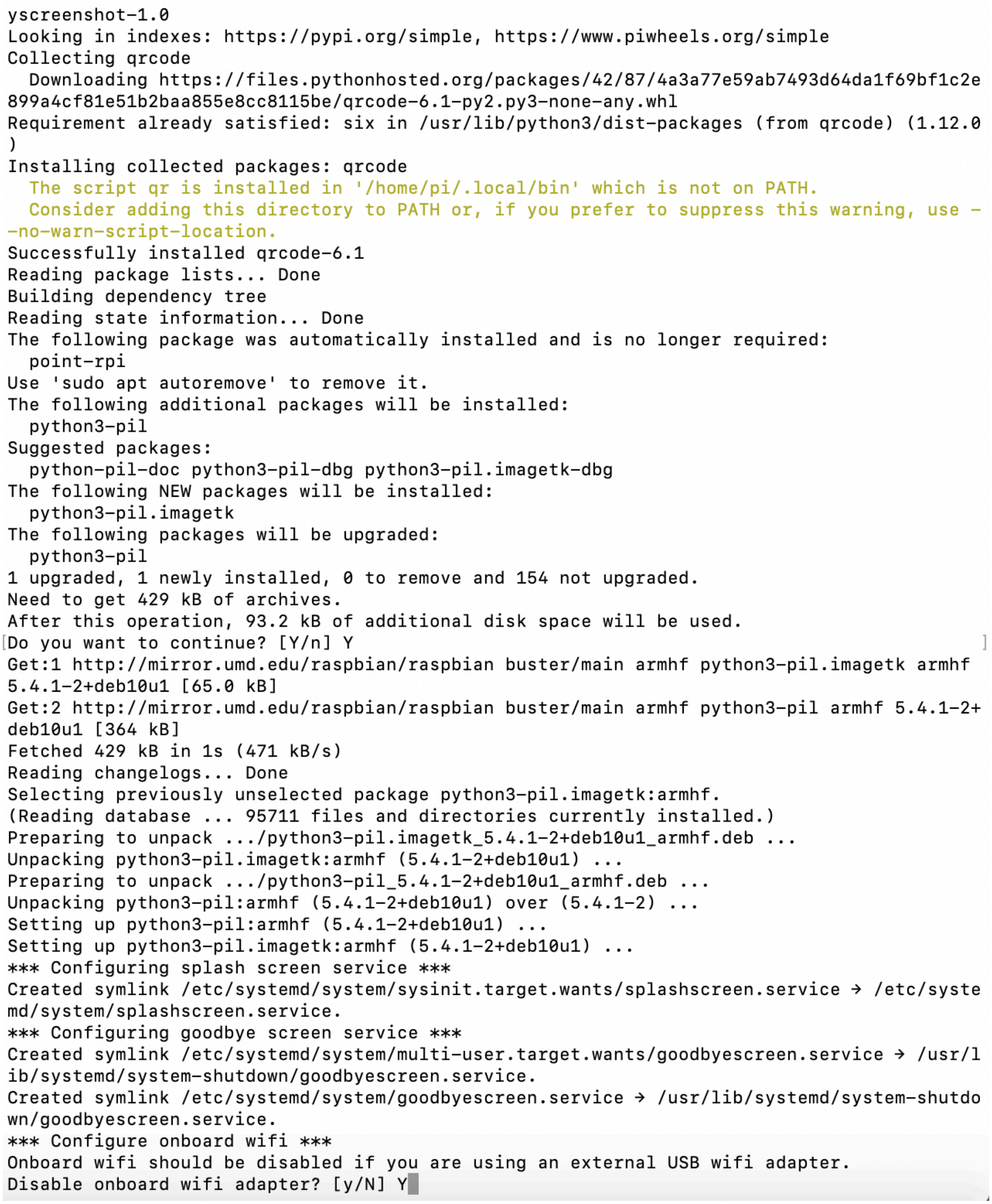
8. Install the Micronets Proto-Pi software by entering the following command:

```
./install
```

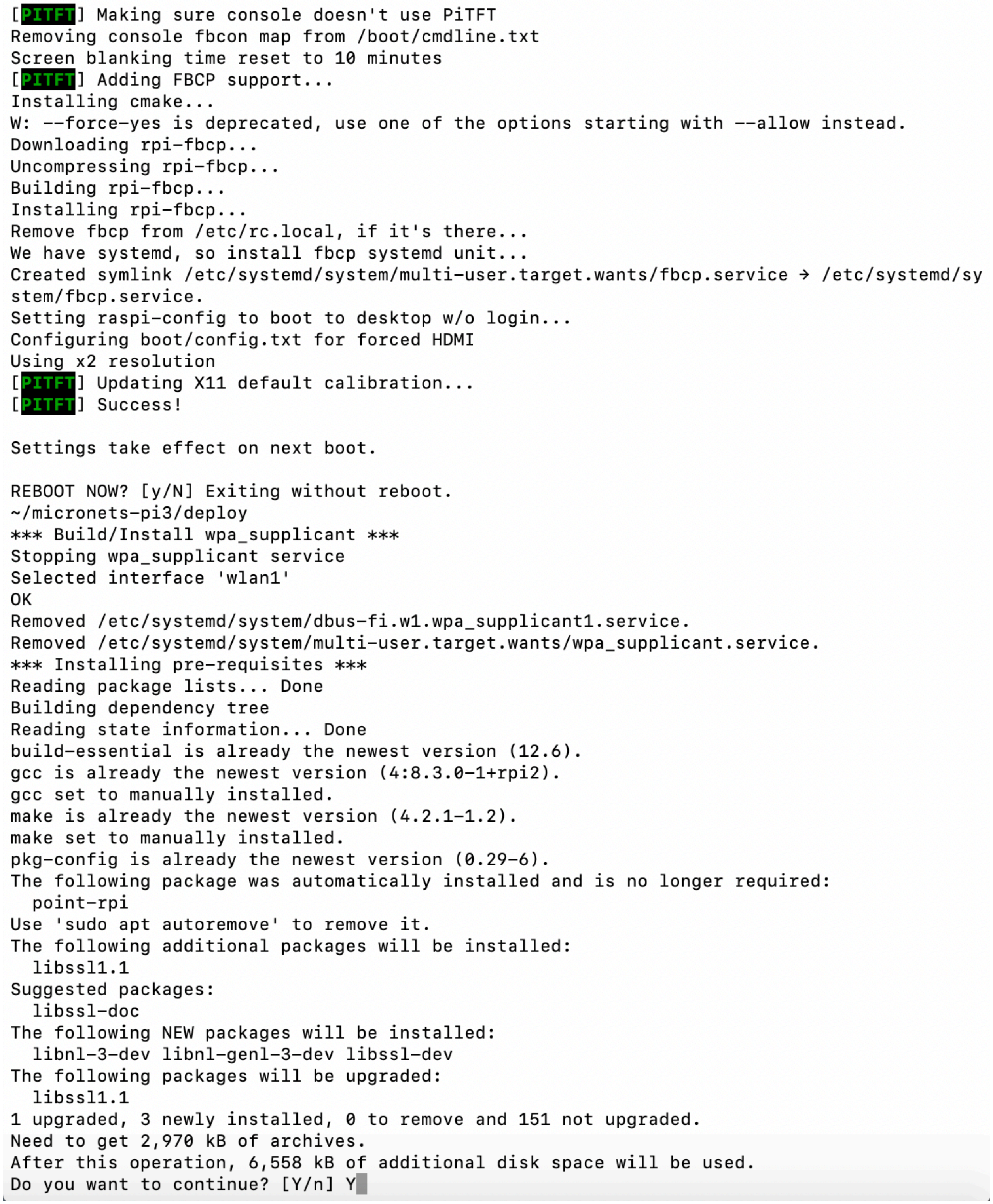
When prompted to accept disk space required, input **Y** as seen below:



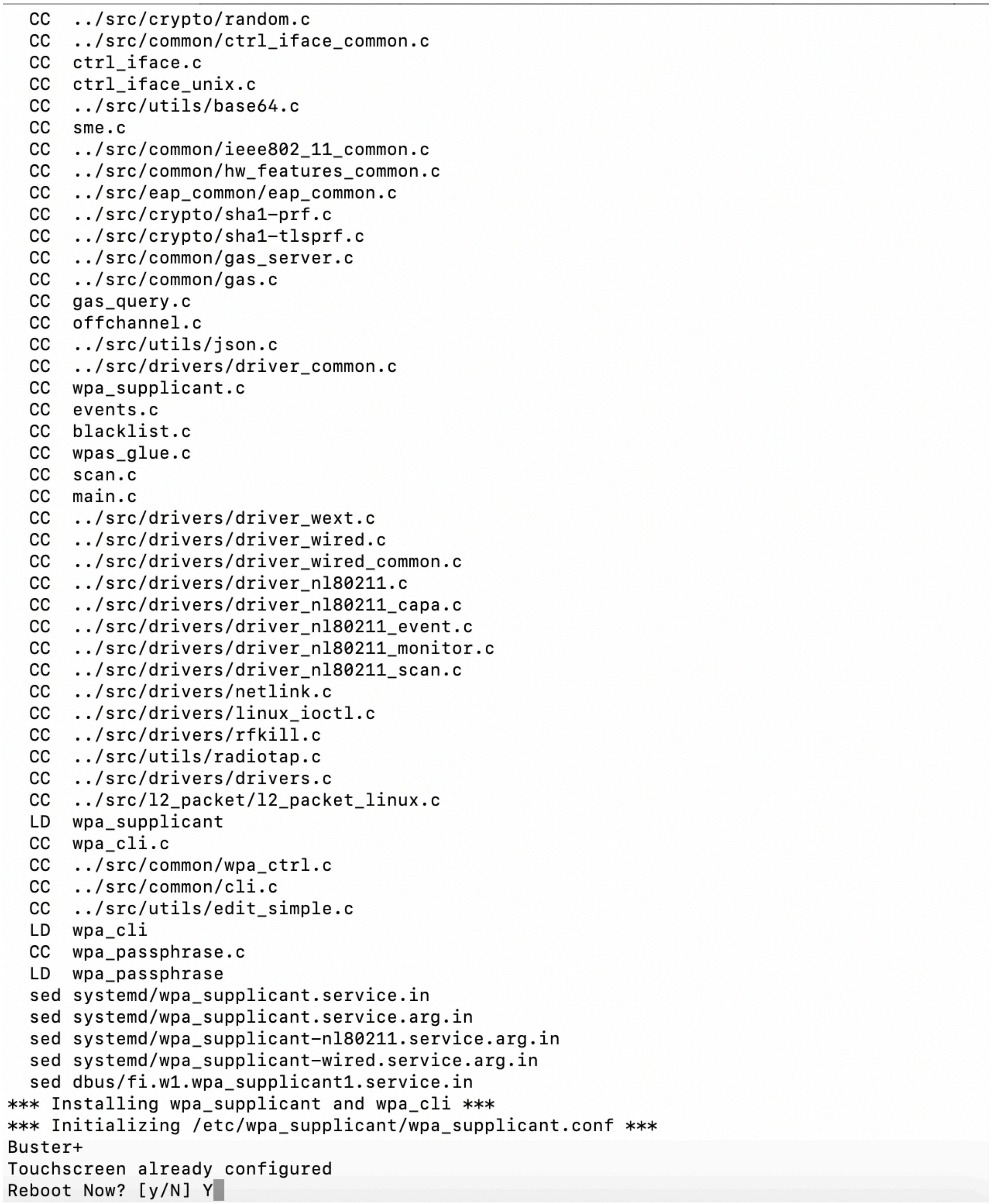
```
Get:4 http://raspbian.raspberrypi.org/raspbian buster/main armhf Packages [13.0 MB]
Fetched 13.4 MB in 13s (1,015 kB/s)
Reading package lists... Done
*** Configuring sudoer privileges required by micronets application (user: pi) ***
*** Adding user pi to groups: netdev, gpio ***
*** Creating desktop autostart file ***
*** Install python (pip3) dependencies ***
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pyscreenshot
  Downloading https://files.pythonhosted.org/packages/ef/f2/35066da41daceabb3d6f1d44d98457
f2b3ddca786181fc7cc9c45e8ef491/pyscreenshot-1.0-py2.py3-none-any.whl
Collecting entrypoint2 (from pyscreenshot)
  Downloading https://files.pythonhosted.org/packages/ca/7e/2c5f211ebbb37c7bd474f3b2d813bd
e5b5391f31c46e190b2b84d83ec9b7/entrypoint2-0.2-py2.py3-none-any.whl
Collecting EasyProcess (from pyscreenshot)
  Downloading https://files.pythonhosted.org/packages/32/8f/88d636f1da22a3c573259e44cfefb4
6a117d3f9432e2c98b1ab4a21372ad/EasyProcess-0.2.10-py2.py3-none-any.whl
Collecting decorator (from entrypoint2->pyscreenshot)
  Downloading https://files.pythonhosted.org/packages/ed/1b/72a1821152d07cf1d8b6fce298aeb0
6a7eb90f4d6d41acec9861e7cc6df0/decorator-4.4.2-py2.py3-none-any.whl
Collecting argparse (from entrypoint2->pyscreenshot)
  Downloading https://files.pythonhosted.org/packages/f2/94/3af39d34be01a24a6e65433d19e107
099374224905f1e0cc6bbe1fd22a2f/argparse-1.4.0-py2.py3-none-any.whl
Installing collected packages: decorator, argparse, entrypoint2, EasyProcess, pyscreenshot
Successfully installed EasyProcess-0.2.10 argparse-1.4.0 decorator-4.4.2 entrypoint2-0.2 p
yscreenshot-1.0
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting qrcode
  Downloading https://files.pythonhosted.org/packages/42/87/4a3a77e59ab7493d64da1f69bf1c2e
899a4cf81e51b2baa855e8cc8115be/qrcode-6.1-py2.py3-none-any.whl
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from qrcode) (1.12.0
)
Installing collected packages: qrcode
  The script qr is installed in '/home/pi/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use -
-no-warn-script-location.
Successfully installed qrcode-6.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  point-rpi
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  python3-pil
Suggested packages:
  python-pil-doc python3-pil-dbg python3-pil.imagetk-dbg
The following NEW packages will be installed:
  python3-pil.imagetk
The following packages will be upgraded:
  python3-pil
1 upgraded, 1 newly installed, 0 to remove and 154 not upgraded.
Need to get 429 kB of archives.
After this operation, 93.2 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```



```
yscreenshot-1.0
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting qrcode
  Downloading https://files.pythonhosted.org/packages/42/87/4a3a77e59ab7493d64da1f69bf1c2e
899a4cf81e51b2baa855e8cc8115be/qrcode-6.1-py2.py3-none-any.whl
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from qrcode) (1.12.0
)
Installing collected packages: qrcode
  The script qr is installed in '/home/pi/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use -
-no-warn-script-location.
Successfully installed qrcode-6.1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  point-rpi
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  python3-pil
Suggested packages:
  python-pil-doc python3-pil-dbg python3-pil.imagetk-dbg
The following NEW packages will be installed:
  python3-pil.imagetk
The following packages will be upgraded:
  python3-pil
1 upgraded, 1 newly installed, 0 to remove and 154 not upgraded.
Need to get 429 kB of archives.
After this operation, 93.2 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://mirror.umd.edu/raspbian/raspbian buster/main armhf python3-pil.imagetk armhf
5.4.1-2+deb10u1 [65.0 kB]
Get:2 http://mirror.umd.edu/raspbian/raspbian buster/main armhf python3-pil armhf 5.4.1-2+
deb10u1 [364 kB]
Fetched 429 kB in 1s (471 kB/s)
Reading changelogs... Done
Selecting previously unselected package python3-pil.imagetk:armhf.
(Reading database ... 95711 files and directories currently installed.)
Preparing to unpack .../python3-pil.imagetk_5.4.1-2+deb10u1_armhf.deb ...
Unpacking python3-pil.imagetk:armhf (5.4.1-2+deb10u1) ...
Preparing to unpack .../python3-pil_5.4.1-2+deb10u1_armhf.deb ...
Unpacking python3-pil:armhf (5.4.1-2+deb10u1) over (5.4.1-2) ...
Setting up python3-pil:armhf (5.4.1-2+deb10u1) ...
Setting up python3-pil.imagetk:armhf (5.4.1-2+deb10u1) ...
*** Configuring splash screen service ***
Created symlink /etc/systemd/system/sysinit.target.wants/splashscreen.service → /etc/syste
md/system/splashscreen.service.
*** Configuring goodbye screen service ***
Created symlink /etc/systemd/system/multi-user.target.wants/goodbyescreen.service → /usr/l
ib/systemd/system-shutdown/goodbyescreen.service.
Created symlink /etc/systemd/system/goodbyescreen.service → /usr/lib/systemd/system-shutdo
wn/goodbyescreen.service.
*** Configure onboard wifi ***
Onboard wifi should be disabled if you are using an external USB wifi adapter.
Disable onboard wifi adapter? [y/N] Y
```



```
[PITFT] Making sure console doesn't use PiTFT
Removing console fbcon map from /boot/cmdline.txt
Screen blanking time reset to 10 minutes
[PITFT] Adding FBCP support...
Installing cmake...
W: --force-yes is deprecated, use one of the options starting with --allow instead.
Downloading rpi-fbcp...
Uncompressing rpi-fbcp...
Building rpi-fbcp...
Installing rpi-fbcp...
Remove fbcp from /etc/rc.local, if it's there...
We have systemd, so install fbcp systemd unit...
Created symlink /etc/systemd/system/multi-user.target.wants/fbcp.service → /etc/systemd/sy
stem/fbcp.service.
Setting raspi-config to boot to desktop w/o login...
Configuring boot/config.txt for forced HDMI
Using x2 resolution
[PITFT] Updating X11 default calibration...
[PITFT] Success!

Settings take effect on next boot.

REBOOT NOW? [y/N] Exiting without reboot.
~/micronets-pi3/deploy
*** Build/Install wpa_supplicant ***
Stopping wpa_supplicant service
Selected interface 'wlan1'
OK
Removed /etc/systemd/system/dbus-fi.w1.wpa_supplicant1.service.
Removed /etc/systemd/system/multi-user.target.wants/wpa_supplicant.service.
*** Installing pre-requisites ***
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.6).
gcc is already the newest version (4:8.3.0-1+rpi2).
gcc set to manually installed.
make is already the newest version (4.2.1-1.2).
make set to manually installed.
pkg-config is already the newest version (0.29-6).
The following package was automatically installed and is no longer required:
  point-rpi
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libssl1.1
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libnl-3-dev libnl-genl-3-dev libssl-dev
The following packages will be upgraded:
  libssl1.1
1 upgraded, 3 newly installed, 0 to remove and 151 not upgraded.
Need to get 2,970 kB of archives.
After this operation, 6,558 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```



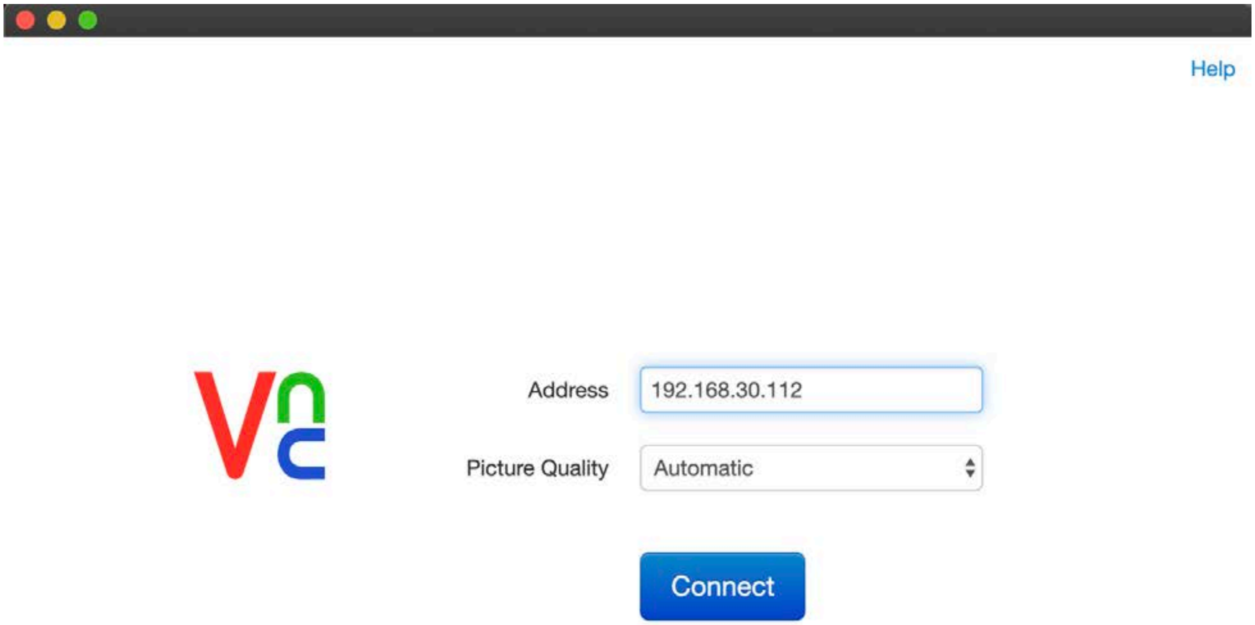
```
  CC  ../src/crypto/random.c
  CC  ../src/common/ctrl_iface_common.c
  CC  ctrl_iface.c
  CC  ctrl_iface_unix.c
  CC  ../src/utils/base64.c
  CC  sme.c
  CC  ../src/common/ieee802_11_common.c
  CC  ../src/common/hw_features_common.c
  CC  ../src/eap_common/eap_common.c
  CC  ../src/crypto/sha1-prf.c
  CC  ../src/crypto/sha1-tlsprf.c
  CC  ../src/common/gas_server.c
  CC  ../src/common/gas.c
  CC  gas_query.c
  CC  offchannel.c
  CC  ../src/utils/json.c
  CC  ../src/drivers/driver_common.c
  CC  wpa_supplicant.c
  CC  events.c
  CC  blacklist.c
  CC  wpas_glue.c
  CC  scan.c
  CC  main.c
  CC  ../src/drivers/driver_wext.c
  CC  ../src/drivers/driver_wired.c
  CC  ../src/drivers/driver_wired_common.c
  CC  ../src/drivers/driver_nl80211.c
  CC  ../src/drivers/driver_nl80211_capa.c
  CC  ../src/drivers/driver_nl80211_event.c
  CC  ../src/drivers/driver_nl80211_monitor.c
  CC  ../src/drivers/driver_nl80211_scan.c
  CC  ../src/drivers/netlink.c
  CC  ../src/drivers/linux_ioctl.c
  CC  ../src/drivers/rfkill.c
  CC  ../src/utils/radiotap.c
  CC  ../src/drivers/drivers.c
  CC  ../src/l2_packet/l2_packet_linux.c
  LD  wpa_supplicant
  CC  wpa_cli.c
  CC  ../src/common/wpa_ctrl.c
  CC  ../src/common/cli.c
  CC  ../src/utils/edit_simple.c
  LD  wpa_cli
  CC  wpa_passphrase.c
  LD  wpa_passphrase
  sed systemd/wpa_supplicant.service.in
  sed systemd/wpa_supplicant.service.arg.in
  sed systemd/wpa_supplicant-nl80211.service.arg.in
  sed systemd/wpa_supplicant-wired.service.arg.in
  sed dbus/fi.w1.wpa_supplicant1.service.in
*** Installing wpa_supplicant and wpa_cli ***
*** Initializing /etc/wpa_supplicant/wpa_supplicant.conf ***
Buster+
Touchscreen already configured
Reboot Now? [y/N] Y
```

4.1.5.3.3 Operation

Four buttons are used for general operation in the Micronets Proto-Pi application. These buttons are on the right side of the application and will be described in the upcoming sections.

1. Accessing Raspberry Pi Using Virtual Network Computing (VNC)Viewer:

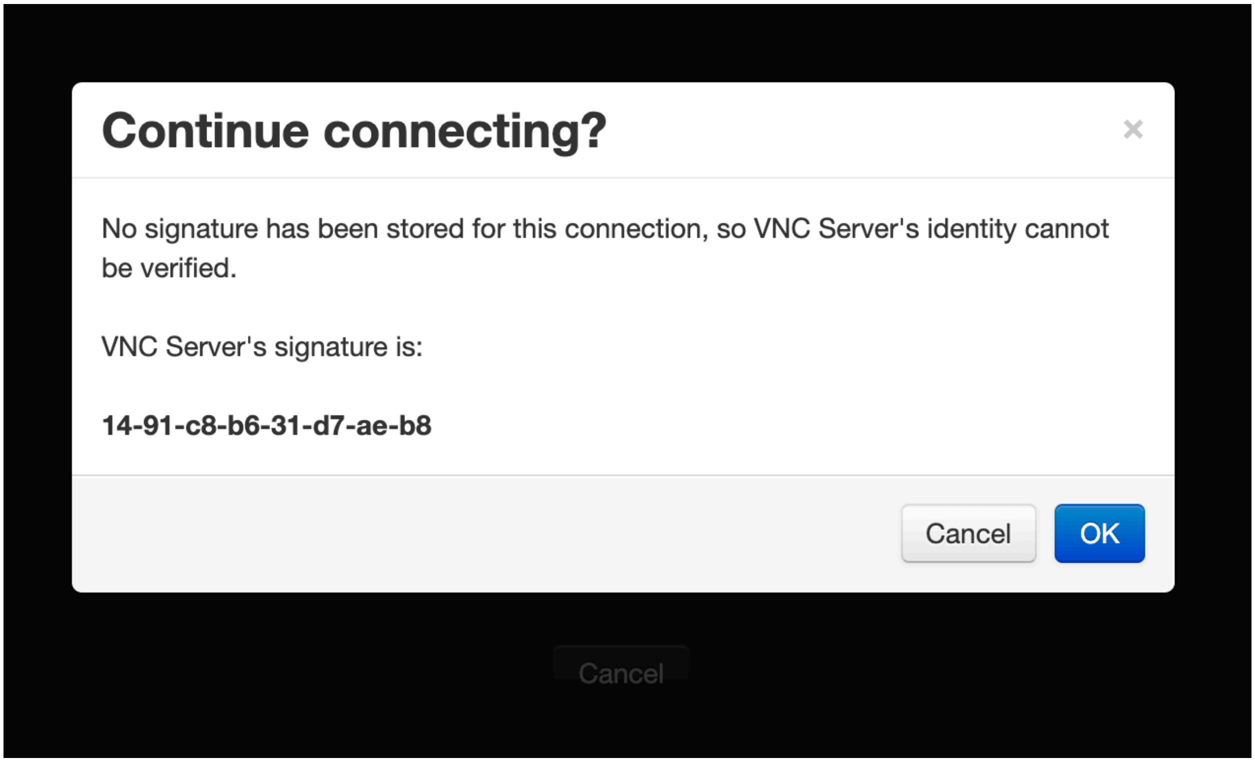
   a. Access the Raspberry Pi using the VNC Viewer, enter the IP address of the Raspberry Pi, and click **Connect:**



Address: 192.168.30.112

Picture Quality: Automatic

You will be prompted to accept and store the signature for this device as it is the first time connecting to it. Click **OK:**



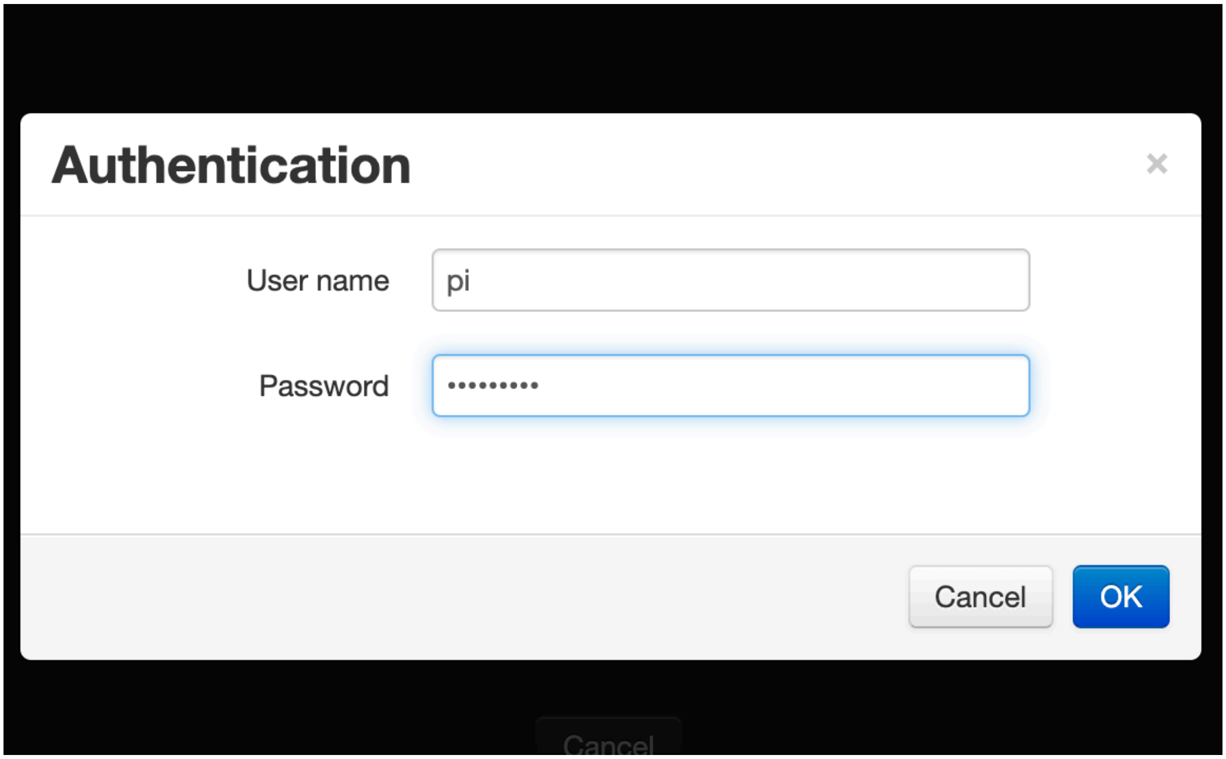
Continue connecting?

No signature has been stored for this connection, so VNC Server's identity cannot be verified.

VNC Server's signature is:

14-91-c8-b6-31-d7-ae-b8

2102 Once accepted, proceed to log in with the username and password, as seen below:

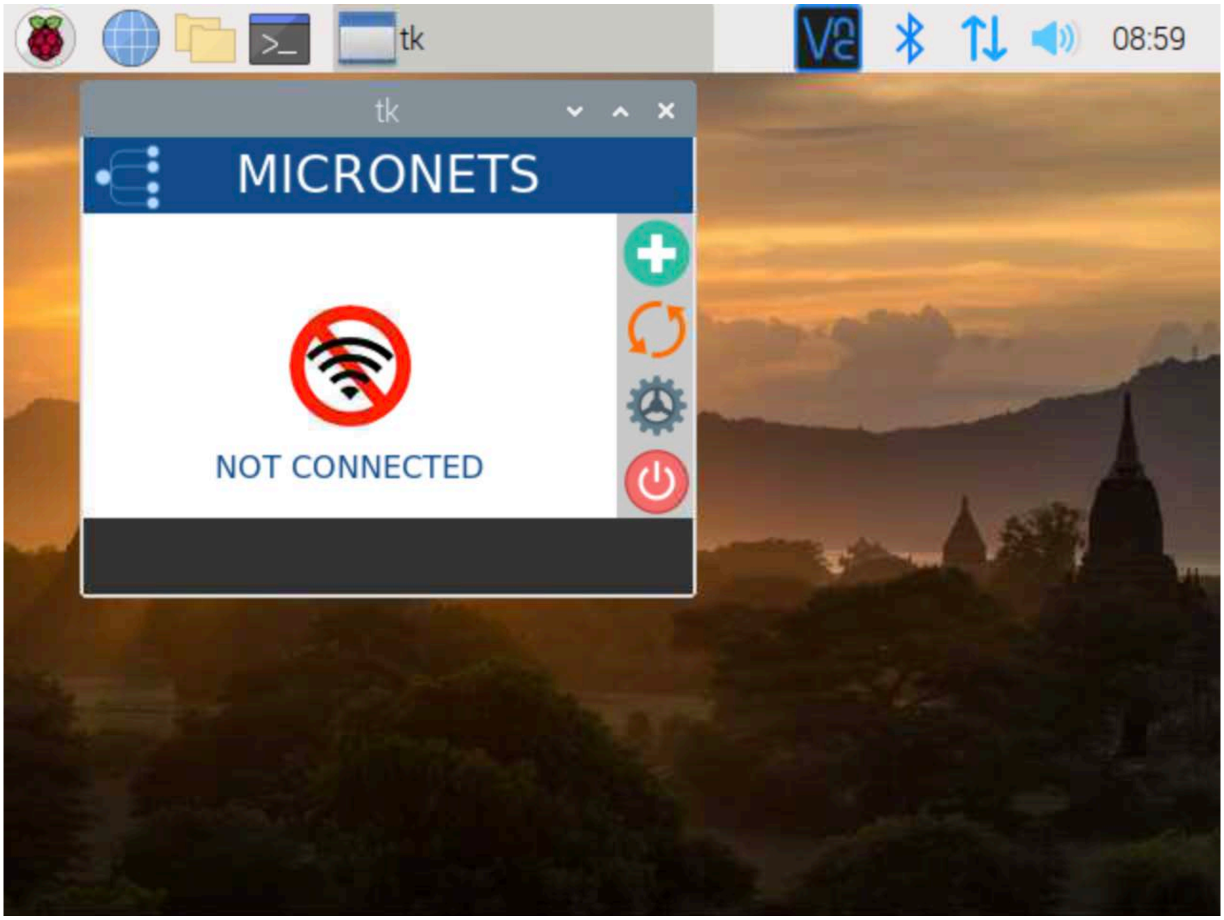


Authentication

User name: pi

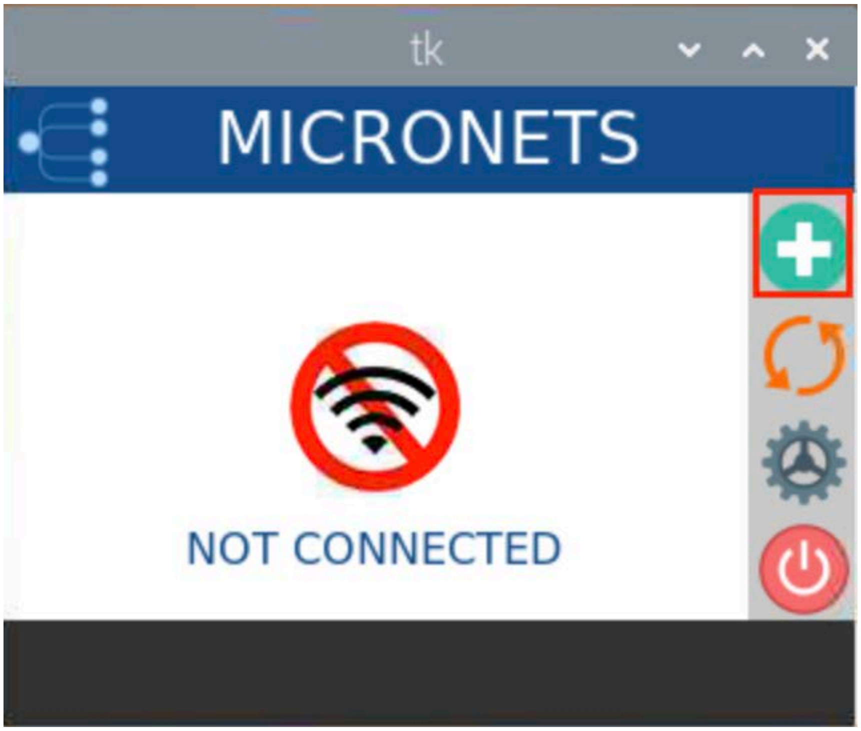
Password: ••••••••••

b. You should see the Micronets Proto-Pi application on the screen as seen below:

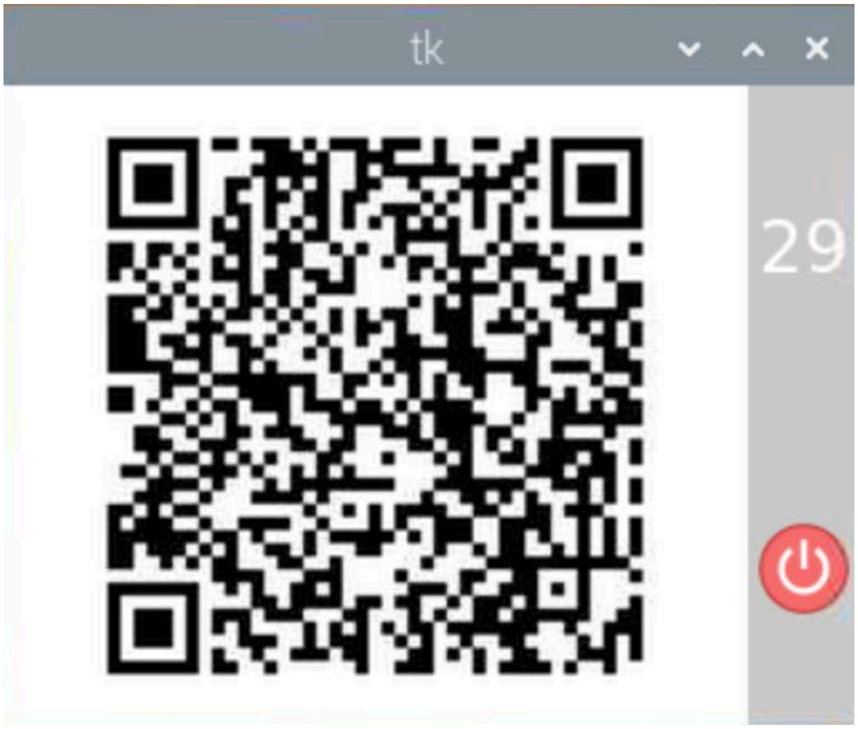


2104 2105 2. The onboard button described in the following steps allows the user to initiate the onboard operation:

2106 a. Click the green button to initiate the onboard process:

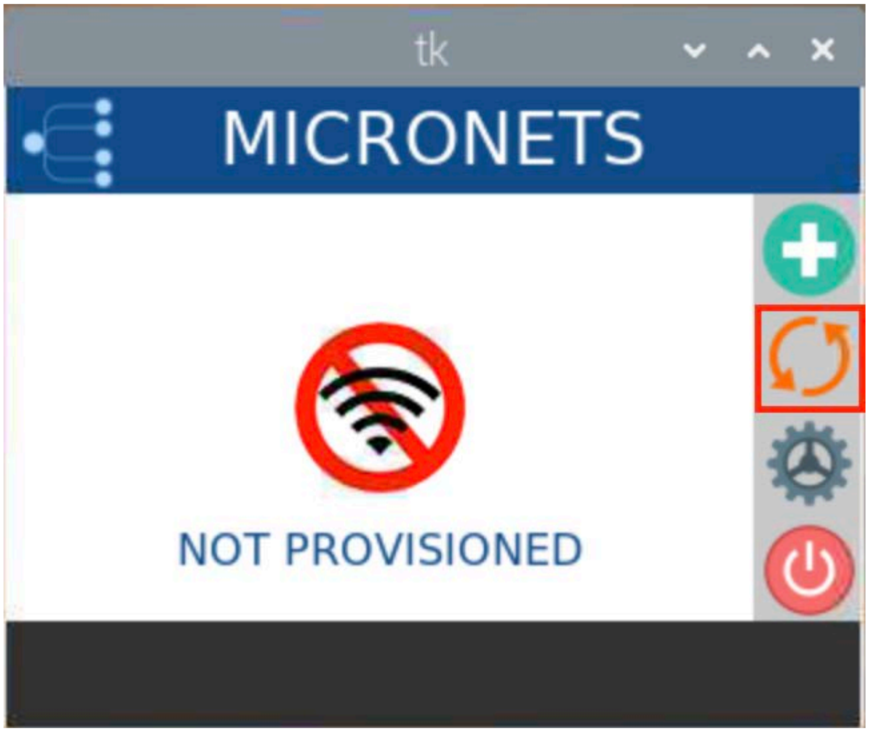


A QR code will appear as seen below. The mobile application will be used to scan this QR code for onboarding:

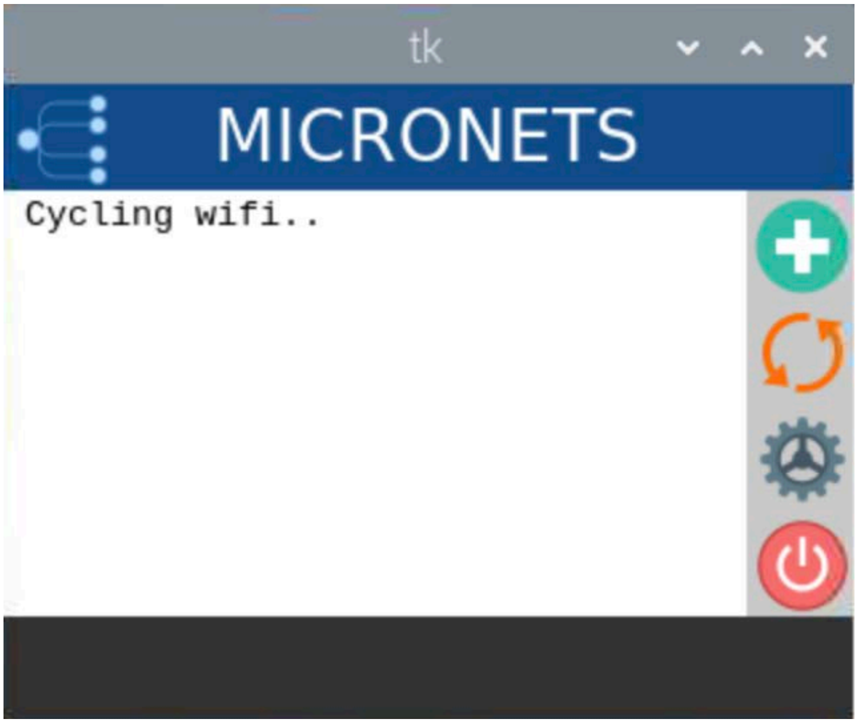


3. The cycle button described in the following steps turns the Wi-Fi off/on to reconnect to the configured service set identifier (SSID).

    a. Click the orange cycle button:

2114 

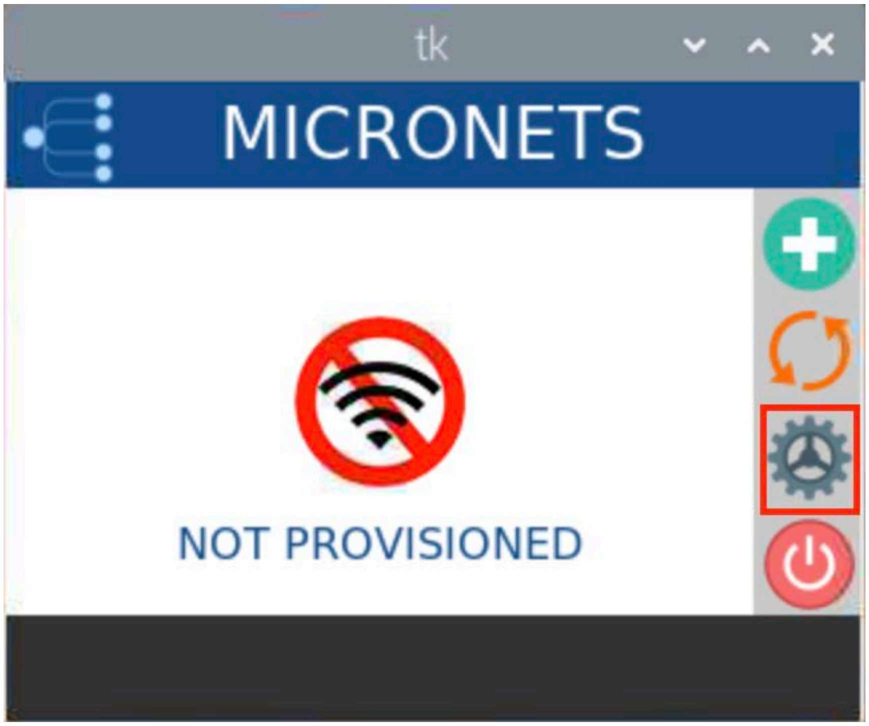
2115 You should see output similar to the following:

2116 

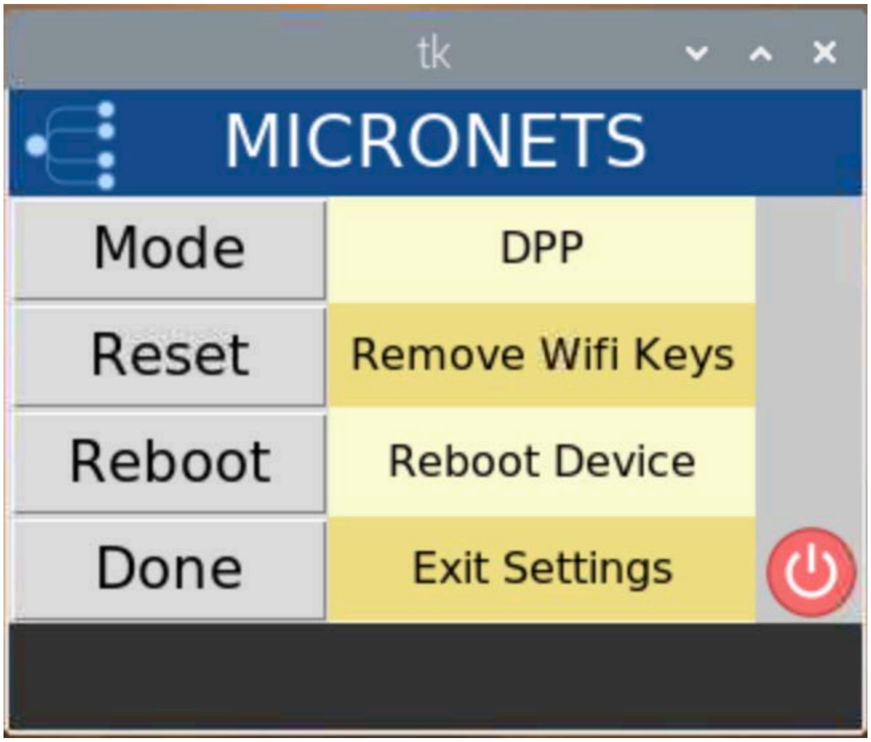
```
Cycling wifi..
```

4. The settings button described in the following steps will open the settings menu, which has four different operations/buttons:

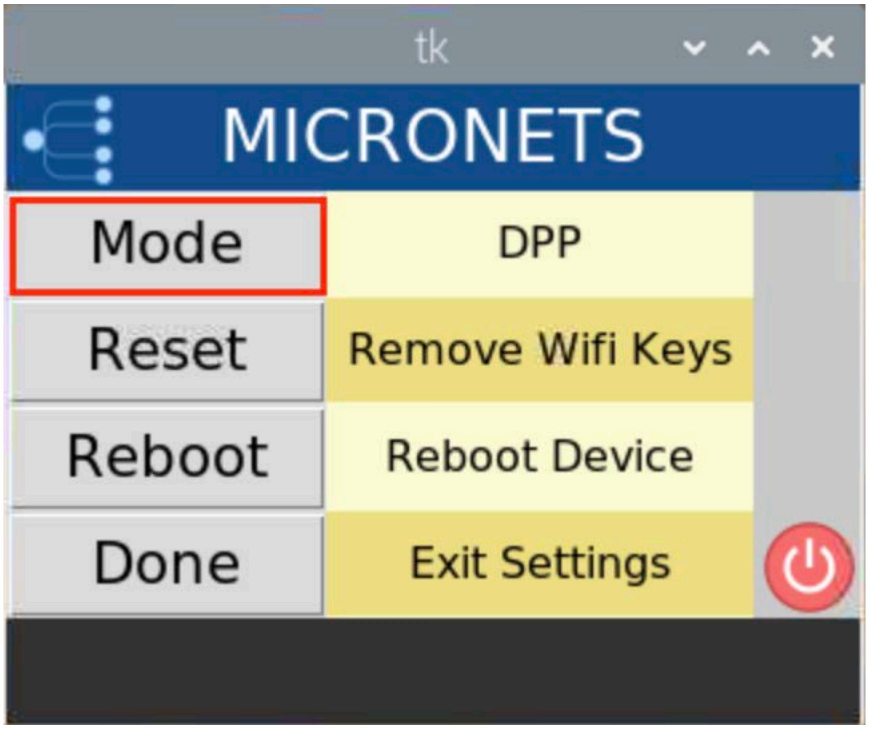
   a. Click the gear button:



The following menu will appear:

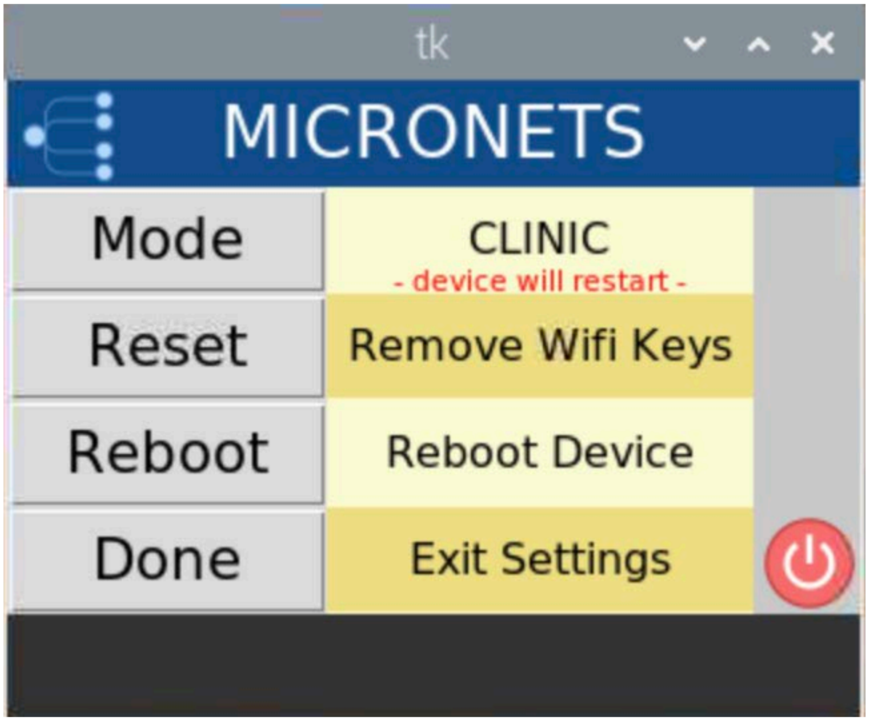


b. Click the **Mode** button to change the onboarding mode from DPP to clinic, and vice versa:



Mode DPP

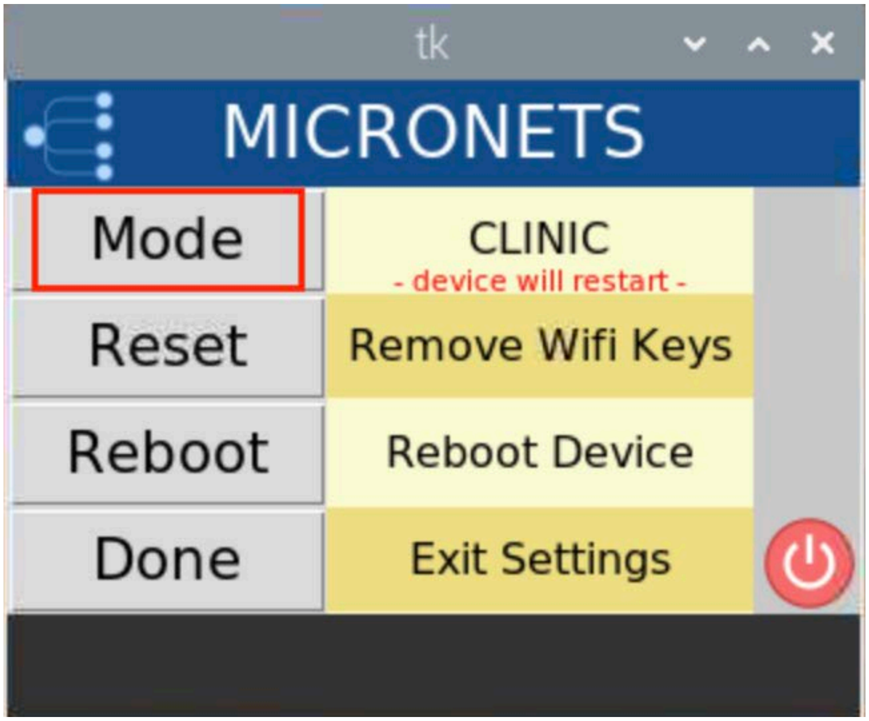
The following screen displays:



Mode CLINIC

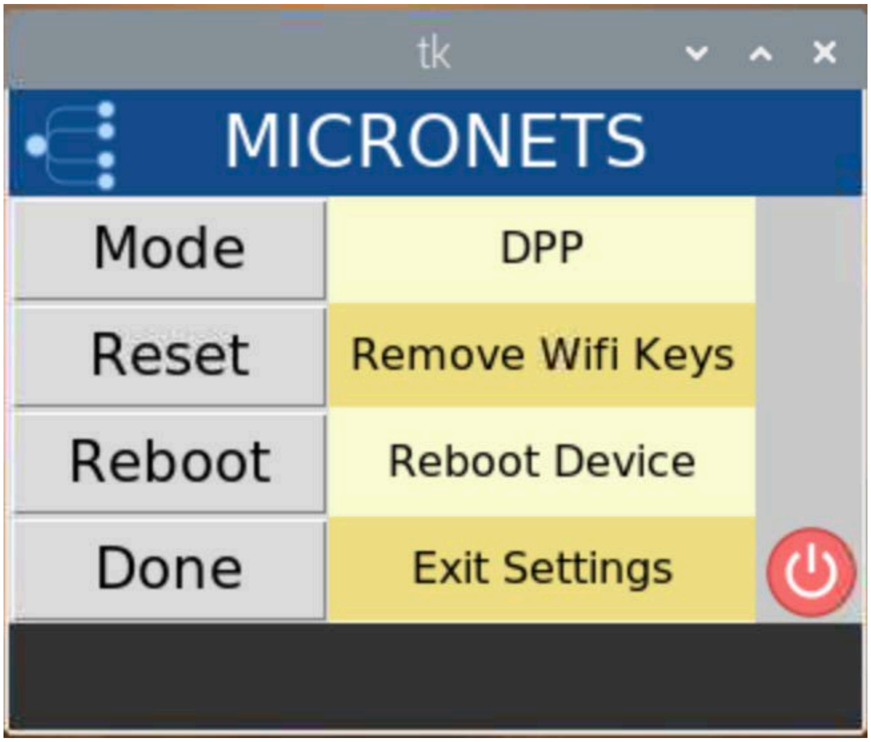
- device will restart -

2128 c. Click the **Mode** button again to return to DPP mode:

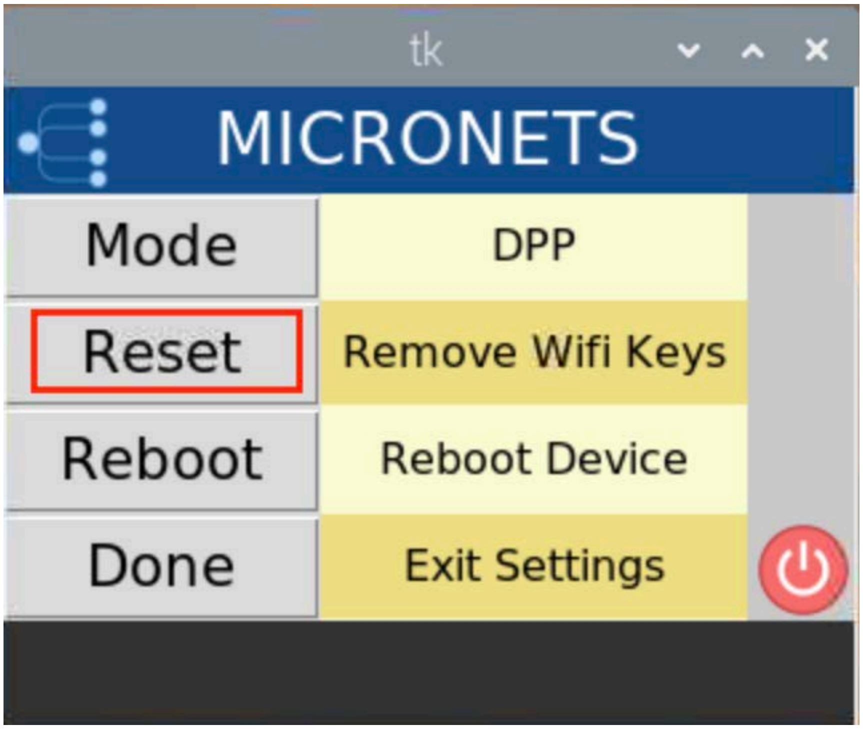


2129

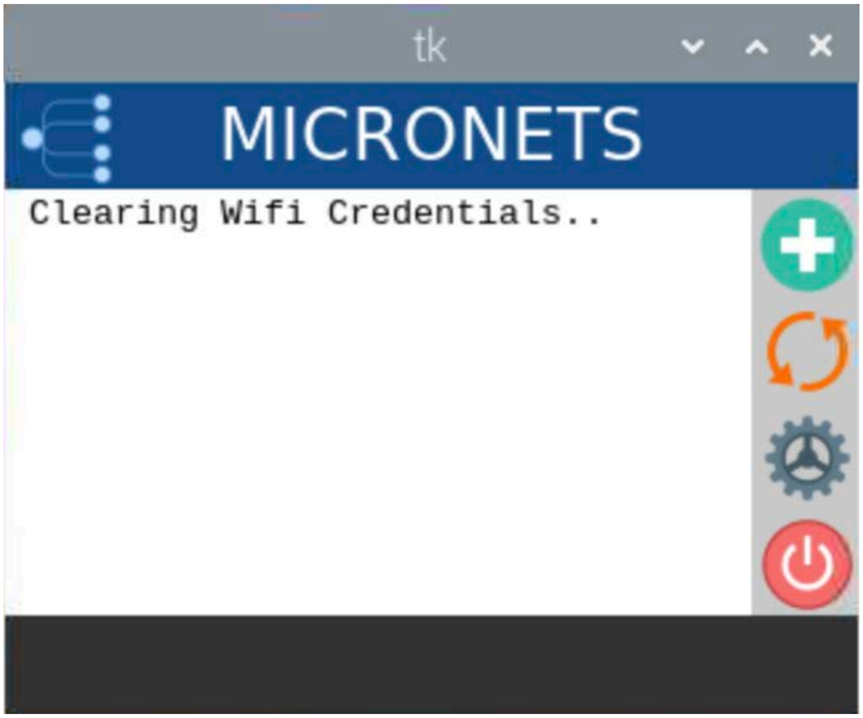
2130 You will see the following change to your screen:



d. Click the **Reset** button to clear Wi-Fi credentials (Note: If the device is in clinic mode, it will restore the credentials for the clinic Wi-Fi):

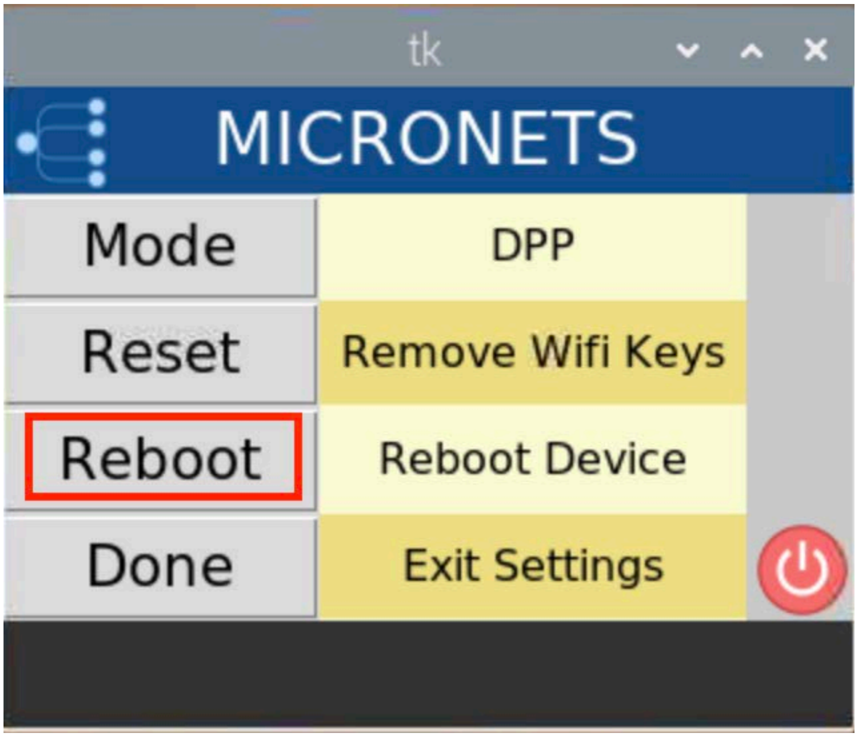


You should see output similar to the following:

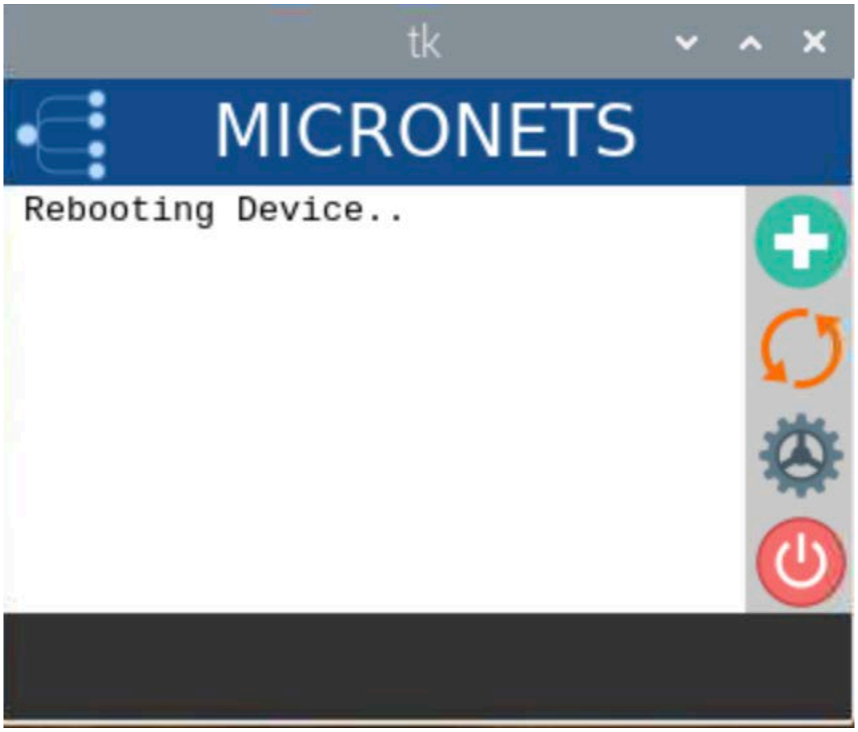


```
Clearing Wifi Credentials..
```

e. Click the **Reboot** button to reboot the Pi:



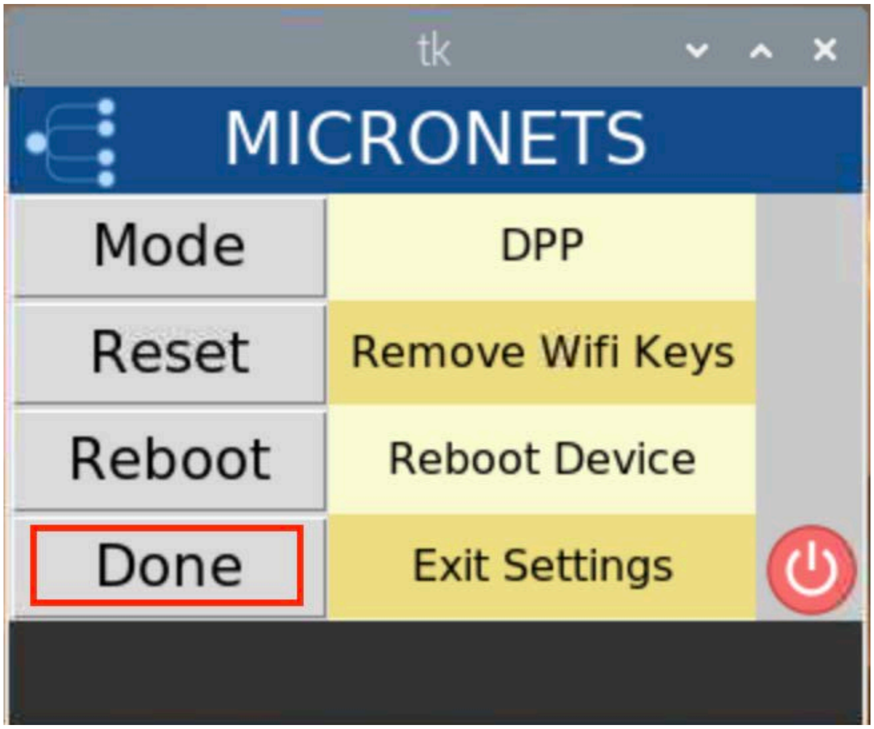
You should see output similar to the following:



Rebooting Device..

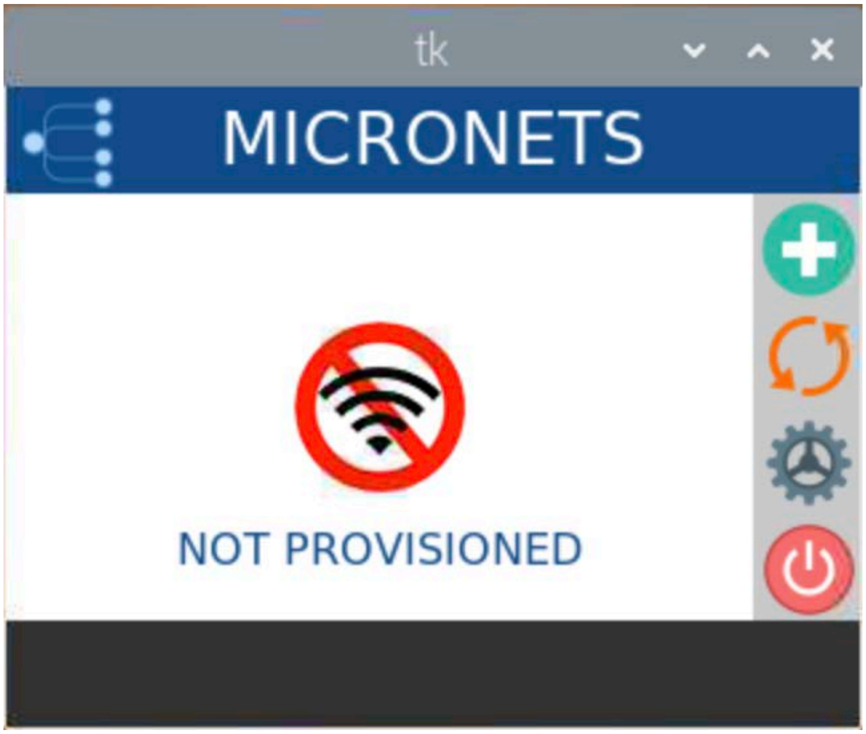
2140

2141 f. Click the **Done** button to exit the settings screen:



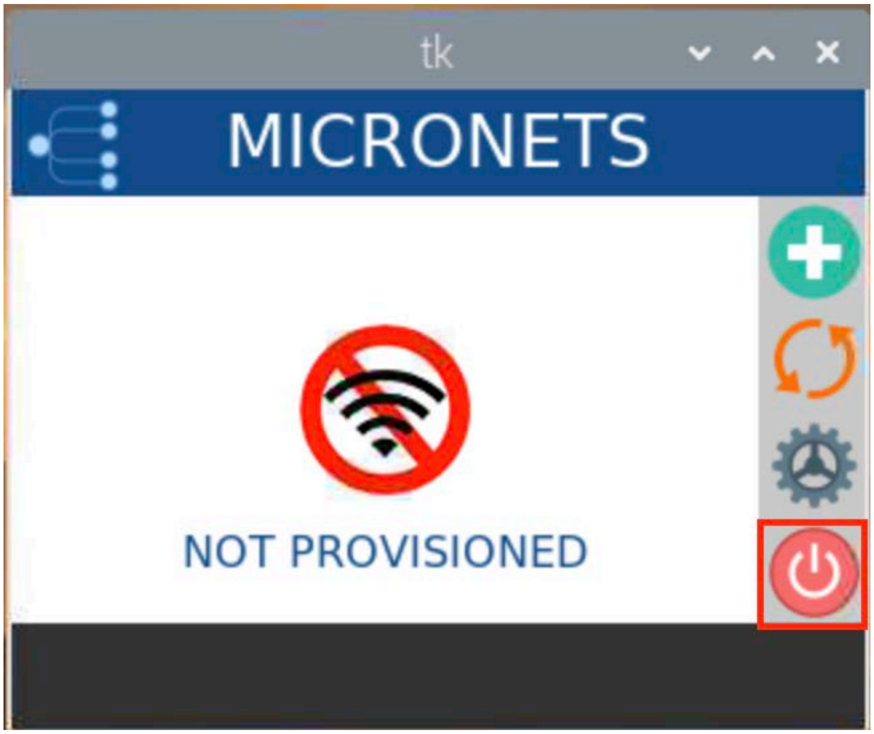
2142

You should see output similar to the following:



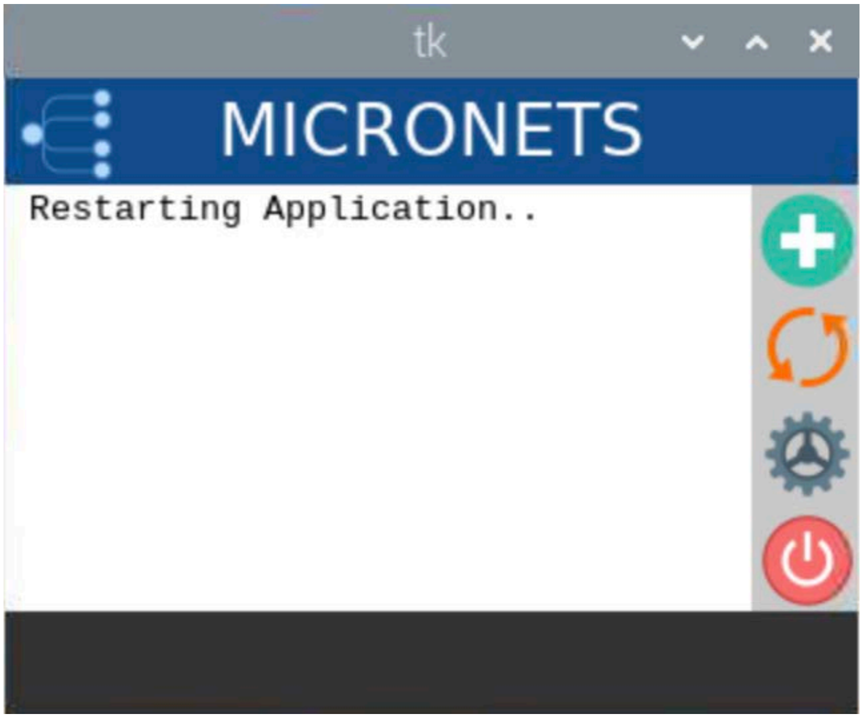
5. The power button described in the following steps appears on the main screen of the Micronets Proto-Pi application and is used to restart the application as well as shut down the Pi entirely:

   a. Tap the power button to restart the application:



2148

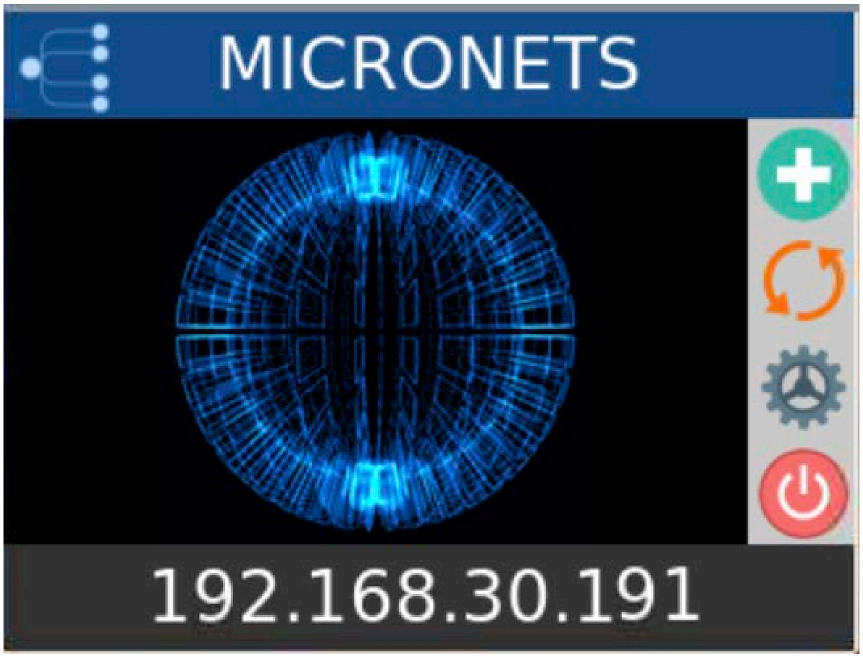
2149 You should see output similar to the following:



```
Restarting Application..
```

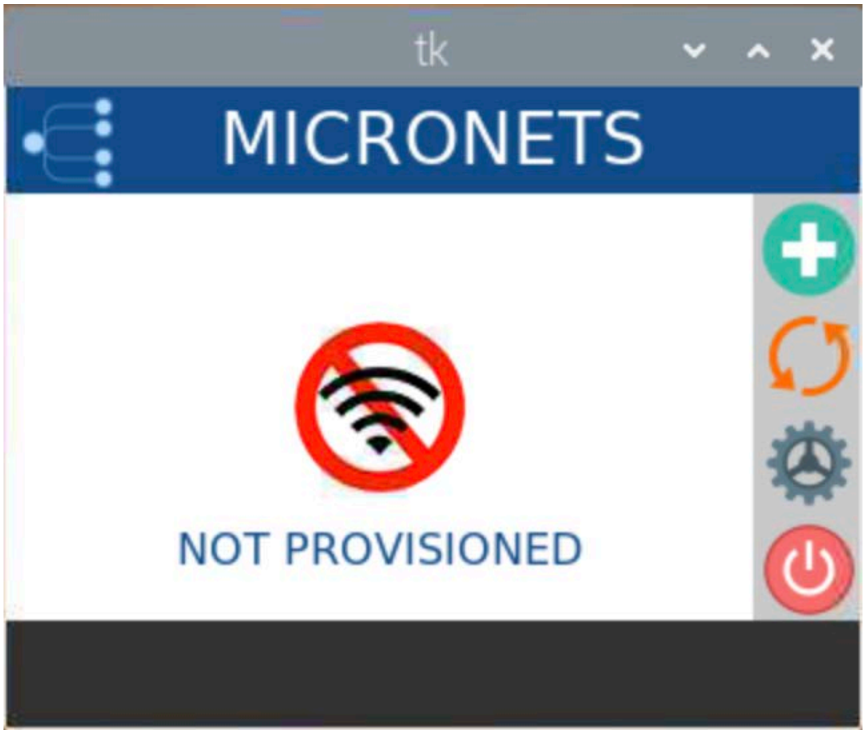
2150

2151 Next, the following screen should appear:



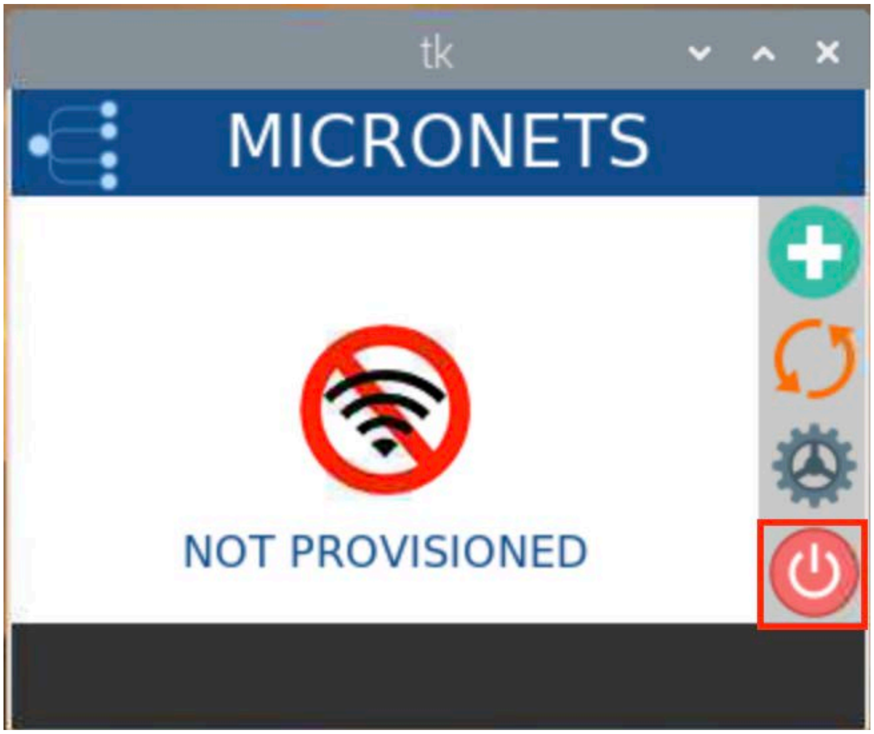
2152

2153 Finally, the main screen appears as seen below:

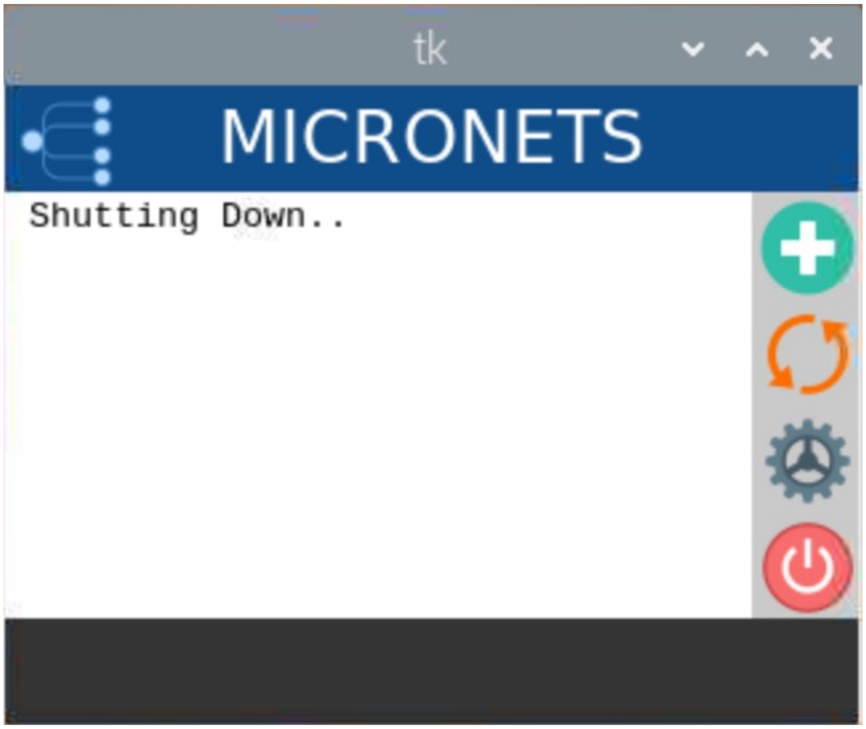


2154

b. Hold the power button to shut down the Pi:



You should see output similar to the following:



Shutting Down..

4.1.6 Update Server

Build 3 leverages the preexisting update server that is described in Build 1's Update Server section. To implement a server that will act as an update server, see the documentation in Build 1's Update Server section. The update server will attempt to access and be accessed by the IoT device, which, in this case, is one of the development kits we built in the lab.

4.1.7 Unapproved Server

Build 3 leverages the preexisting unapproved server that is described in Build 1's Unapproved Server section. To implement a server that will act as an unapproved server, see the documentation in Build 1's Unapproved Server section. The unapproved server will attempt to access and be accessed by an IoT device, which, in this case, is one of the MUD-capable devices on the implementation network.

4.1.8 CableLabs MUD Registry

This section describes the CableLabs MUD registry, which, for this implementation, is a cloud-provided service. This implementation leveraged the nccoe-build-3 branch of CableLabs MUD registry Git release. This service can be hosted by the implementer or another party. This documentation describes setting up your own MUD registry.

4.1.8.1 CableLabs MUD Registry Overview

The Micronets MUD registry provides the capability to look up the MUD URL that is associated with a particular device. This registration and MUD URL association can be done manually or by the device using self-registration.

4.1.8.2 Configuration Overview

The following subsections document the software and network configurations for the MUD registry. Please note that the MUD manager, Micronets Manager, Websocket Proxy, MUD registry, and MSO portal are all implemented on the same server, nccoe-server1.micronets.net. Many of these configurations have already been covered in previous sections of this document but are repeated here for consistency.

4.1.8.2.1 Network Configuration

This server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.8.2.2 Software Configuration

For this build, the server ran on an Ubuntu 18.04 LTS operating system. The MUD registry runs in its own docker container and is configured to use SSL/TLS encryption.

The following software is required to install, configure, and operate the MUD registry:

- an Ubuntu 18.04 LTS server reachable by the server hosting the Micronets Manager instances and any Micronets gateways
- docker (v18.06 or higher)
- curl
- NGINX

4.1.8.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the MUD registry:

- 4 GB of RAM
- 50 GB of free disk space

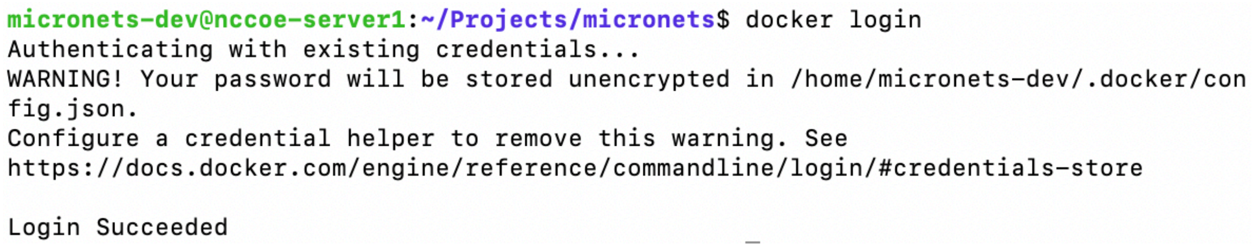
4.1.8.3 Setup

4.1.8.3.1 Install and Configure MUD Registry

1. Log in to docker by using the following command:

   ```
   docker login
   ```

   You should see output similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /home/micronets-dev/.docker/con
fig.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

2. Retrieve the nccoe-build-3 tagged image by entering the following command:

   ```
   docker pull community.cablelabs.com:4567/micronets-docker/micronets-mud-registry:nccoe-build-3
   ```

3. Execute the following command to run the image that was just retrieved:

   The command will follow the syntax below. Replace **<MUDFILESERVER_URL>** with your MUD file server URL:

   docker run -d -p 127.0.0.1:3082:3082 --env mud_base_uri=https://**<MUDFILESERVER_URL>** -v /etc/micronets/micronets-mud-registry.d/:/etc/micronets/config --name=micronets-mud-registry community.cablelabs.com:4567/micronets-docker/micronets-mud-registry:nccoe-build-3

   ```
   docker run -d -p 127.0.0.1:3082:3082 --env mud_base_uri=https://nccoe-server2.micronets.net/micronets-mud -v /etc/micronets/micronets-mud-registry.d/:/etc/micronets/config --name=micronets-mud-registry community.cablelabs.com:4567/micronets-docker/micronets-mud-registry:nccoe-build-3
   ```

4. Configure your own vendor code for your implementation by completing the following steps:

   a. Create and modify the ***mud-registry.conf*** file by executing the following command. (Note: The configuration file must be named “mud-registry.conf” and must reside in a host folder that is passed to the docker instance in the docker run command executed in the previous step.)

      ```
      sudo vim /etc/micronets/micronets-mud-registry.d/mud-registry.conf
      ```

   b. Replace **<VENDOR-CODE>** with your choice of vendor name, <**MUDREGISTRY_URL>** with the MUD registry URL, and **<MUDFILESERVER_URL>** with the MUD file server URL:

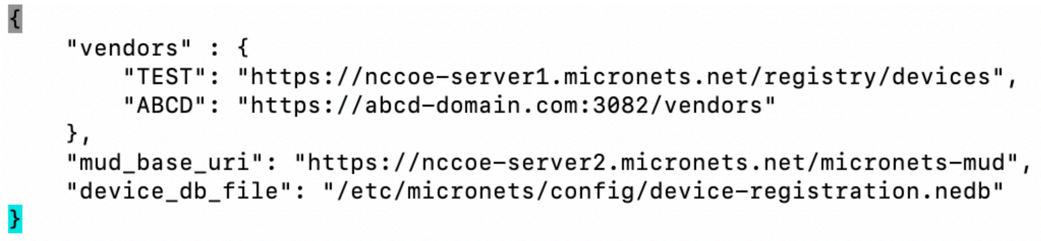
      {

          "vendors" : {

```
            "<VENDOR-CODE> ": "https:// <MUDREGISTRY_URL> /registry/devices",
            "ABCD": "https://abcd-domain.com:3082/vendors"
        },
        "mud_base_uri": "https:// <MUDFILESERVER_URL> /micronets-mud",
        "device_db_file": "/etc/micronets/config/device-registration.nedb"
}
```

For this implementation, we added the following:

```
{
    "vendors" : {
        "TEST": "https://nccoe-server1.micronets.net/registry/devices",
        "ABCD": "https://abcd-domain.com:3082/vendors"
    },
    "mud_base_uri": "https://nccoe-server2.micronets.net/micronets-mud",
    "device_db_file": "/etc/micronets/config/device-registration.nedb"
}
```



```
{
    "vendors" : {
        "TEST": "https://nccoe-server1.micronets.net/registry/devices",
        "ABCD": "https://abcd-domain.com:3082/vendors"
    },
    "mud_base_uri": "https://nccoe-server2.micronets.net/micronets-mud",
    "device_db_file": "/etc/micronets/config/device-registration.nedb"
}
```

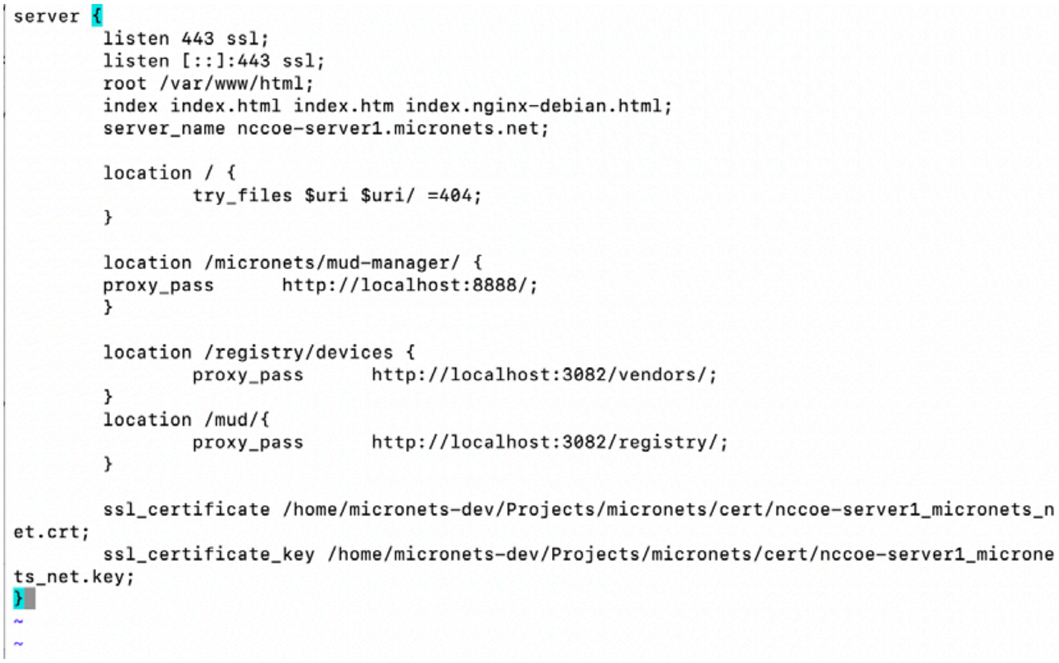
c. Modify the sites-available file for the NGINX server to route appropriate traffic to the docker container by executing the following commands:

   i. Open the sites-available file for the NGINX server by entering the following command:

```
sudo vim /etc/nginx/sites-available/nccoe-server1.micronets.net
```

ii. Map the location for the /registry/devices so it is routed to vendors/ in the docker instance running on port 3082 and for the /mud/ to be passed to the global registry by adding the following to the server block:

```
location /registry/devices {
        proxy_pass    http://localhost:3082/vendors/;
}
location /mud/{
        proxy_pass    http://localhost:3082/registry/;
}
```



```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nccoe-server1.micronets.net;

        location / {
                try_files $uri $uri/ =404;
        }

        location /micronets/mud-manager/ {
        proxy_pass      http://localhost:8888/;
        }

        location /registry/devices {
                proxy_pass      http://localhost:3082/vendors/;
        }
        location /mud/{
                proxy_pass      http://localhost:3082/registry/;
        }

        ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-server1_micronets_n
et.crt;
        ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-server1_microne
ts_net.key;
}
~
~
```

4.1.9 CableLabs Micronets Manager for SDN Control

This section describes the CableLabs Micronets Manager, which, for this implementation, is a cloud-provided service. This implementation leveraged the nccoe-build-3 branch of CableLabs Micronets Manager Git release. This service can be hosted by the implementer or another party. This documentation describes setting up your own Micronets Manager.

4.1.9.1 CableLabs Micronets Manager Overview

The Micronets Manager provides micro-services to the implementation. It receives onboarding requests, bootstrapping information, and more for a particular subscriber and is a core component for handing off requests among different components in the architecture.

4.1.9.2 Configuration Overview

The following subsections document the software and network configurations for the Micronets Manager. Please note that these instructions have the MUD manager, Micronets Manager, Websocket Proxy, MUD registry, and MSO portal all deployed onto the same server, nccoe-server1.micronets.net. Many of these configurations are already covered in previous sections of this document but are repeated here for consistency.

4.1.9.2.1 Network Configuration

This server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.9.2.2 Software Configuration

For this build, the server ran on an Ubuntu 18.04 LTS operating system. The Micronets Manager runs in its own docker container and is configured to use SSL/TLS encryption.

The following software is required to install, configure, and operate the Micronets Manager:

- an Ubuntu 18.04 LTS server reachable by any Micronets gateways
- docker (v18.06 or higher)
- docker-compose (v1.23.1 or higher)
- OpenSSL (1.0.2g or higher)
- curl
- NGINX (1.14.0 or higher)

4.1.9.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the Micronets Manager:

- 4 GB of RAM
- 50 GB of free disk space

4.1.9.3 Setup

4.1.9.3.1 Install Dependencies

1. Install docker, docker-compose, openssl, curl, and NGINX by entering the following command:

```
sudo apt-get install docker docker-compose openssl curl nginx
```

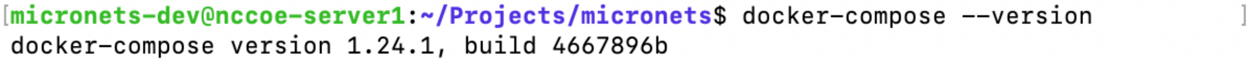
4.1.9.3.2 Install and Configure the Micronets Manager

1. Ensure the version of docker-compose is correct and upgrade if needed:

   a. Check the current version by entering the following command:

   ```
   docker-compose –version
   ```

   You should see the version output as seen below:

   ```
   [micronets-dev@nccoe-server1:~/Projects/micronets$ docker-compose --version
    docker-compose version 1.24.1, build 4667896b
   ```

   b. If the version is earlier than v1.23.1, run the following command to install a new version in /usr/local/bin directory:

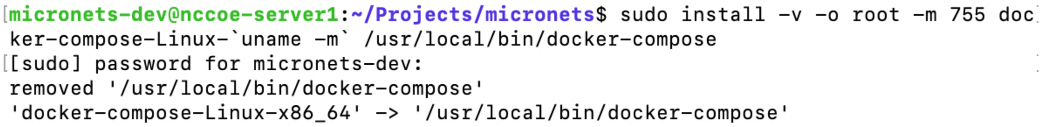
      i. Download the docker-compose utility:

      ```
      curl -s -L -O https://github.com/docker/compose/releases/down-
      load/1.24.1/docker-compose-Linux-`uname -m`
      ```

      ii. Install the docker-compose utility to the appropriate directory:

      ```
      sudo install -v -o root -m 755 docker-compose-Linux-`uname -m`
      /usr/local/bin/docker-compose
      ```

      You should see output similar to the following:

      ```
      [micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 doc
       ker-compose-Linux-`uname -m` /usr/local/bin/docker-compose
      [[sudo] password for micronets-dev:
       removed '/usr/local/bin/docker-compose'
       'docker-compose-Linux-x86_64' -> '/usr/local/bin/docker-compose'
      ```

2. Download the Micronets Manager management script, and install it by entering the following commands:

   a. Download the Micronets Manager management script:

   ```
   curl -s -O https://raw.githubusercontent.com/cablelabs/micronets-man-
   ager/nccoe-build-3/scripts/mm-container
   ```

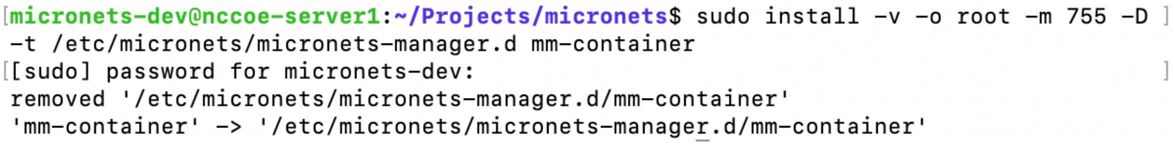
   b. Download the docker-compose utility:

   ```
   curl -s -O https://raw.githubusercontent.com/cablelabs/micronets-man-
   ager/nccoe-build-3/scripts/docker-compose.yml
   ```

c. Install the management script to the appropriate location:

```
sudo install -v -o root -m 755 -D -t /etc/micronets/micronets-manager.d
mm-container
```

You should see output similar to the following:

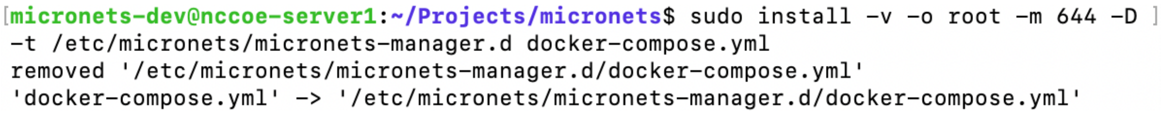


```
micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 -D
-t /etc/micronets/micronets-manager.d mm-container
[sudo] password for micronets-dev:
removed '/etc/micronets/micronets-manager.d/mm-container'
'mm-container' -> '/etc/micronets/micronets-manager.d/mm-container'
```

d. Install the docker-compose utility to the appropriate location:

```
sudo install -v -o root -m 644 -D -t /etc/micronets/micronets-manager.d
docker-compose.yml
```

You should see output similar to the following:



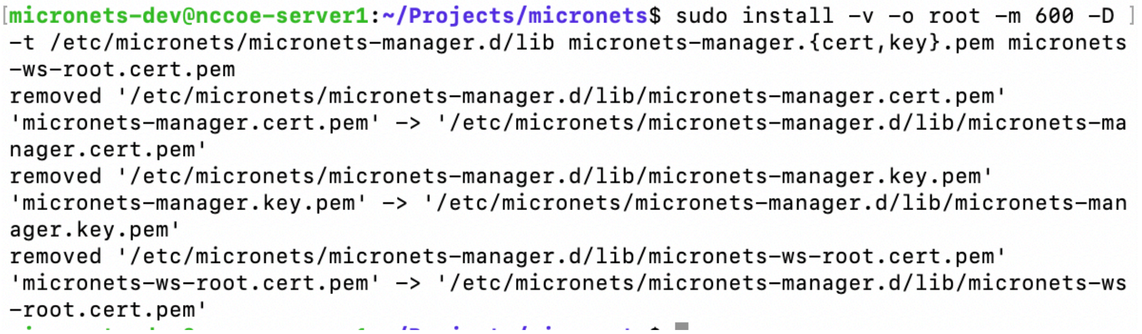
```
micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 644 -D
-t /etc/micronets/micronets-manager.d docker-compose.yml
removed '/etc/micronets/micronets-manager.d/docker-compose.yml'
'docker-compose.yml' -> '/etc/micronets/micronets-manager.d/docker-compose.yml'
```

3. Copy the Micronets Manager server cert/key and the Websocket Proxy root CA cert created in earlier steps for use by the Micronets Manager docker container(s):

   a. Install the certificates and keys by entering the following command:

```
sudo install -v -o root -m 600 -D -t /etc/micronets/micronets-man-
ager.d/lib micronets-manager.{cert,key}.pem micronets-ws-root.cert.pem
```

   You should see output similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 600 -D
-t /etc/micronets/micronets-manager.d/lib micronets-manager.{cert,key}.pem micronets
-ws-root.cert.pem
removed '/etc/micronets/micronets-manager.d/lib/micronets-manager.cert.pem'
'micronets-manager.cert.pem' -> '/etc/micronets/micronets-manager.d/lib/micronets-ma
nager.cert.pem'
removed '/etc/micronets/micronets-manager.d/lib/micronets-manager.key.pem'
'micronets-manager.key.pem' -> '/etc/micronets/micronets-manager.d/lib/micronets-man
ager.key.pem'
removed '/etc/micronets/micronets-manager.d/lib/micronets-ws-root.cert.pem'
'micronets-ws-root.cert.pem' -> '/etc/micronets/micronets-manager.d/lib/micronets-ws
-root.cert.pem'
```

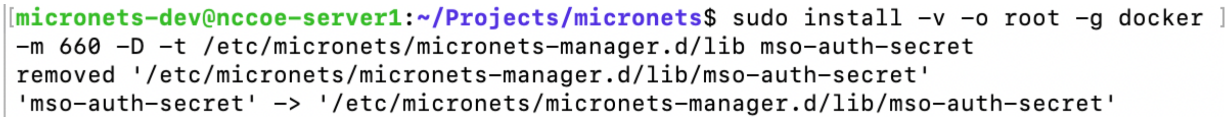
   b. Create a placeholder *micronets-ws-proxy.pkeycert.pem* file. This file is not used, but the Micronets Manager currently checks for it:

```
sudo touch /etc/micronets/micronets-manager.d/lib/micronets-ws-
proxy.pkeycert.pem
```

4. Copy the shared secret value generated during the MSO portal installation:

```
sudo install -v -o root -g docker -m 660 -D -t /etc/micronets/micronets-
manager.d/lib mso-auth-secret
```

You should see output similar to the following:

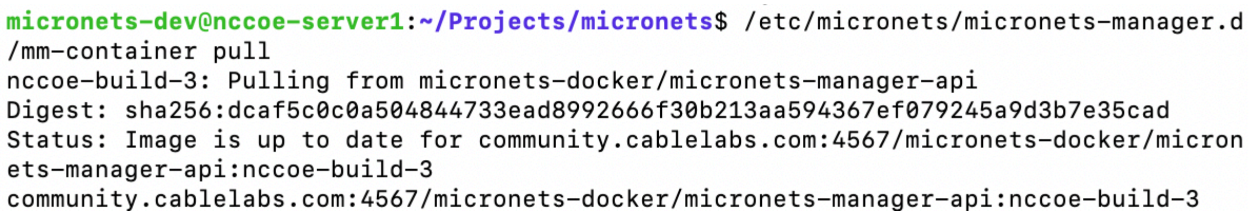


```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -g docker ]
-m 660 -D -t /etc/micronets/micronets-manager.d/lib mso-auth-secret
removed '/etc/micronets/micronets-manager.d/lib/mso-auth-secret'
'mso-auth-secret' -> '/etc/micronets/micronets-manager.d/lib/mso-auth-secret'
```

5. Execute the following command to download the Micronets Manager docker image (Note: If you cannot connect to the docker service, use sudo usermod -aG docker to add the user account to the docker group):

```
/etc/micronets/micronets-manager.d/mm-container pull
```

You should see output similar to the following:



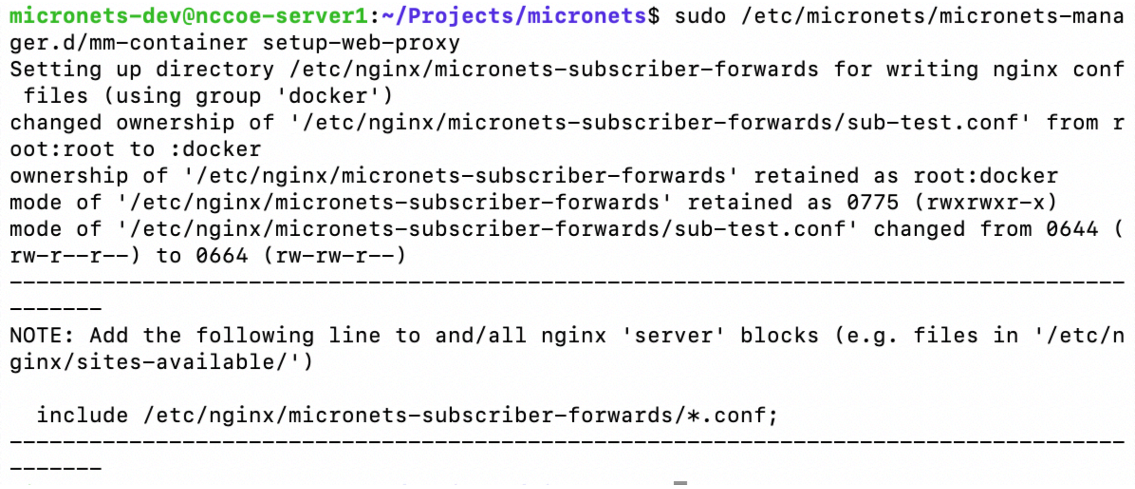
```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-manager.d
/mm-container pull
nccoe-build-3: Pulling from micronets-docker/micronets-manager-api
Digest: sha256:dcaf5c0c0a504844733ead8992666f30b213aa594367ef079245a9d3b7e35cad
Status: Image is up to date for community.cablelabs.com:4567/micronets-docker/micron
ets-manager-api:nccoe-build-3
community.cablelabs.com:4567/micronets-docker/micronets-manager-api:nccoe-build-3
```

6. Complete the following steps to configure NGINX for the Micronets Manager:

   d. The Micronets Manager management script creates NGINX forward entries that provide a unique URI for each Micronets Manager docker image. To create the infrastructure for these entries, run:

   ```
   sudo /etc/micronets/micronets-manager.d/mm-container setup-web-proxy
   ```

   You should see output similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ sudo /etc/micronets/micronets-mana
ger.d/mm-container setup-web-proxy
Setting up directory /etc/nginx/micronets-subscriber-forwards for writing nginx conf
 files (using group 'docker')
changed ownership of '/etc/nginx/micronets-subscriber-forwards/sub-test.conf' from r
oot:root to :docker
ownership of '/etc/nginx/micronets-subscriber-forwards' retained as root:docker
mode of '/etc/nginx/micronets-subscriber-forwards' retained as 0775 (rwxrwxr-x)
mode of '/etc/nginx/micronets-subscriber-forwards/sub-test.conf' changed from 0644 (
rw-r--r--) to 0664 (rw-rw-r--)
------------------------------------------------------------------------------------
-------
NOTE: Add the following line to and/all nginx 'server' blocks (e.g. files in '/etc/n
ginx/sites-available/')

  include /etc/nginx/micronets-subscriber-forwards/*.conf;
------------------------------------------------------------------------------------
-------
```

7. This sets up the folder to dynamically create forwarding entries for Micronets Manager instances as they are created/removed. But the site files in /etc/nginx/sites-available/ need the following added to the server blocks to enable forwarding subscriber operations to the correct docker container.

    a. Open the NGINX sites-available file created in:

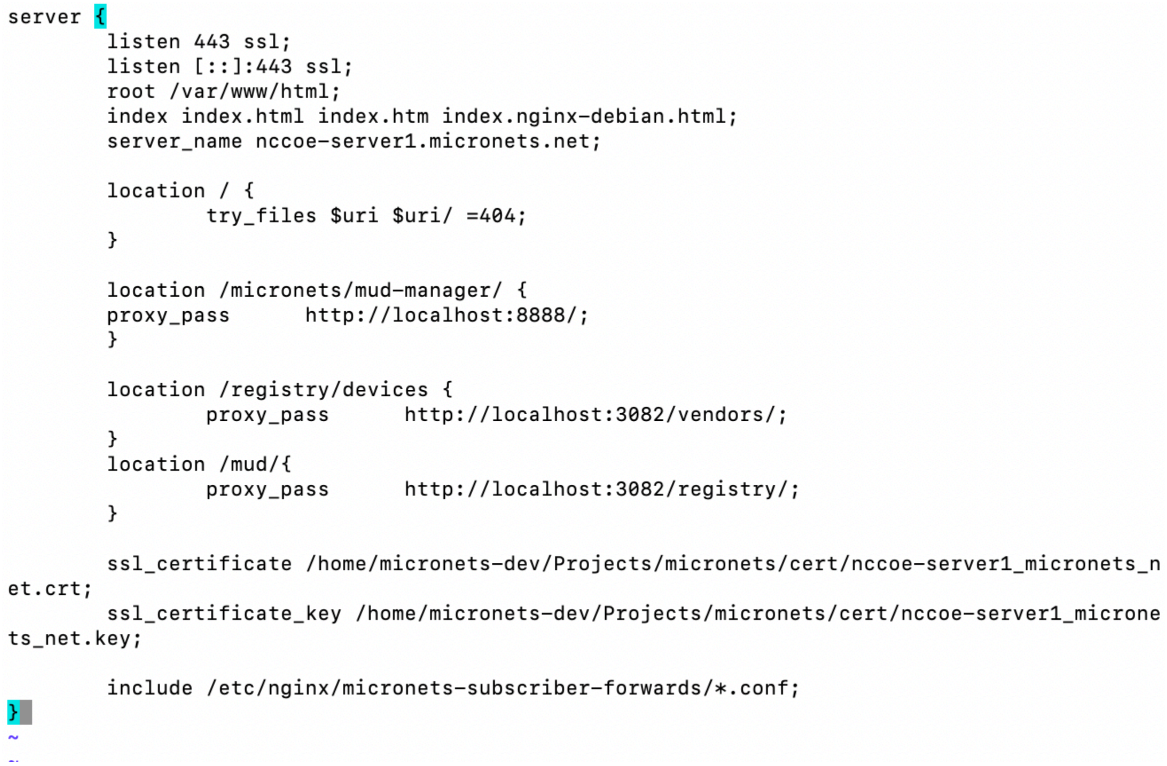
    ```
    sudo vim /etc/nginx/sites-available/nccoe-server1.micronets.net
    ```

    b. Add the following entry to the file:

    ```
    include /etc/nginx/micronets-subscriber-forwards/*.conf;
    ```

    For example:

    ```
    server {
        server_name nccoe-server1.micronets.net;
     {…]
        include /etc/nginx/micronets-subscriber-forwards/*.conf;
    }
    ```



```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nccoe-server1.micronets.net;

        location / {
                try_files $uri $uri/ =404;
        }

        location /micronets/mud-manager/ {
        proxy_pass      http://localhost:8888/;
        }

        location /registry/devices {
                proxy_pass      http://localhost:3082/vendors/;
        }
        location /mud/{
                proxy_pass      http://localhost:3082/registry/;
        }

        ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-server1_micronets_n
et.crt;
        ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-server1_microne
ts_net.key;

        include /etc/nginx/micronets-subscriber-forwards/*.conf;
}
~
```

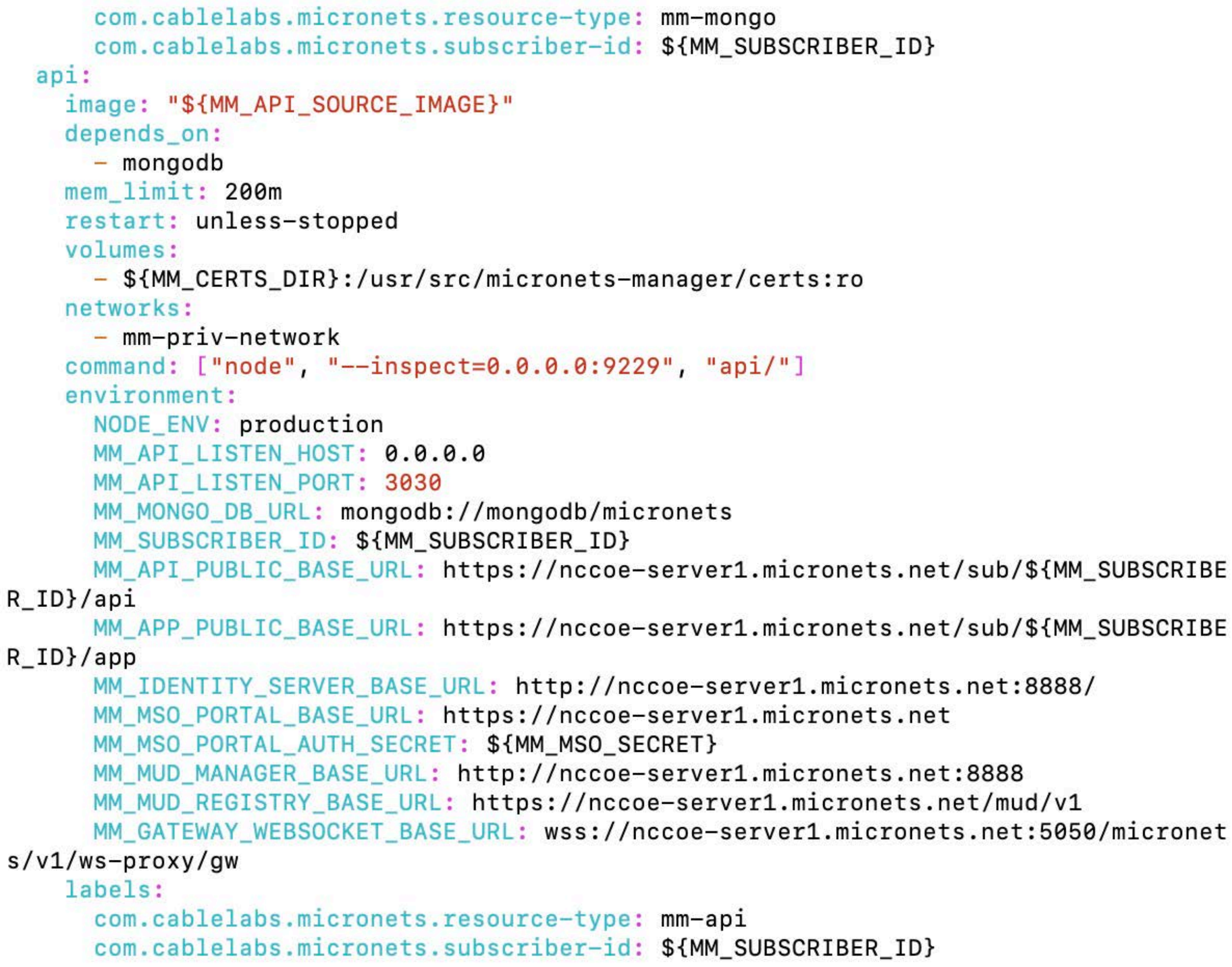
8. Complete the following steps to configure the Micronets Manager to communicate with other Micronets services on the server:

    a. Open the ***docker-compose.yml*** file by entering the following command:

    ```
    sudo vim /etc/micronets/micronets-manager.d/docker-compose.yml
    ```

    b. Modify the following environmental variables in the *docker-compose.yml* file. Replace **<ServerURL>** with your server URL:

    ```
     MM_API_PUBLIC_BASE_URL: https://<ServerURL>/sub/${MM_SUBSCRIBER_ID}/api
     MM_APP_PUBLIC_BASE_URL: https:// <ServerURL>/sub/${MM_SUBSCRIBER_ID}/app
     MM_IDENTITY_SERVER_BASE_URL: https://<ServerURL>:8888/
     MM_MSO_PORTAL_BASE_URL: https:// <ServerURL>/micronets/mso-portal
     MM_MUD_MANAGER_BASE_URL: https:// <ServerURL>/micronets/mud-manager
     MM_MUD_REGISTRY_BASE_URL: https:// <ServerURL>/micronets/mud/v1
     MM_GATEWAY_WEBSOCKET_BASE_URL: wss://<ServerURL>:5050/micronets/v1/ws-
    proxy/gw
    ```



```
      com.cablelabs.micronets.resource-type: mm-mongo
      com.cablelabs.micronets.subscriber-id: ${MM_SUBSCRIBER_ID}
  api:
    image: "${MM_API_SOURCE_IMAGE}"
    depends_on:
      - mongodb
    mem_limit: 200m
    restart: unless-stopped
    volumes:
      - ${MM_CERTS_DIR}:/usr/src/micronets-manager/certs:ro
    networks:
      - mm-priv-network
    command: ["node", "--inspect=0.0.0.0:9229", "api/"]
    environment:
      NODE_ENV: production
      MM_API_LISTEN_HOST: 0.0.0.0
      MM_API_LISTEN_PORT: 3030
      MM_MONGO_DB_URL: mongodb://mongodb/micronets
      MM_SUBSCRIBER_ID: ${MM_SUBSCRIBER_ID}
      MM_API_PUBLIC_BASE_URL: https://nccoe-server1.micronets.net/sub/${MM_SUBSCRIBE
R_ID}/api
      MM_APP_PUBLIC_BASE_URL: https://nccoe-server1.micronets.net/sub/${MM_SUBSCRIBE
R_ID}/app
      MM_IDENTITY_SERVER_BASE_URL: http://nccoe-server1.micronets.net:8888/
      MM_MSO_PORTAL_BASE_URL: https://nccoe-server1.micronets.net
      MM_MSO_PORTAL_AUTH_SECRET: ${MM_MSO_SECRET}
      MM_MUD_MANAGER_BASE_URL: http://nccoe-server1.micronets.net:8888
      MM_MUD_REGISTRY_BASE_URL: https://nccoe-server1.micronets.net/mud/v1
      MM_GATEWAY_WEBSOCKET_BASE_URL: wss://nccoe-server1.micronets.net:5050/micronet
s/v1/ws-proxy/gw
    labels:
      com.cablelabs.micronets.resource-type: mm-api
      com.cablelabs.micronets.subscriber-id: ${MM_SUBSCRIBER_ID}
```

4.1.10 Micronets Websocket Proxy

This section describes the CableLabs Micronets Websocket Proxy, which, for this implementation, is a cloud-provided service. This implementation leverages the nccoe-build-3 branch of CableLabs Micronets Websocket Proxy Git release. This service can be hosted by the implementer or another party. This documentation describes setting up your own Micronets Manager.

4.1.10.1 Micronets Websocket Proxy Overview

The Micronets Websocket Proxy is a service for establishing a Websocket connection between a subscriber's gateway and Micronets Manager. This connection is leveraged to issue representational state transfer (REST) commands to the gateway and to receive event notifications from the gateway.

4.1.10.2 Configuration Overview

The following subsections document the software and network configurations for the Websocket Proxy. Please note that the MUD manager, Micronets Manager, Websocket Proxy, MUD registry, and MSO portal are all implemented on the same server, nccoe-server1.micronets.net. Many of these configurations are already covered in previous sections of this document but are repeated here for consistency.

4.1.10.2.1 Network Configuration

This server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.10.2.2 Software Configuration

For this build, the server ran on an Ubuntu 18.04 LTS operating system. The Websocket Proxy runs in its own docker container and is configured to use SSL/TLS encryption.

The following software is required to install, configure, and operate the Websocket Proxy:

- an Ubuntu 18.04 LTS server reachable by the Micronets Manager and any Micronets gateways
- docker (v18.06 or higher)
- docker-compose (v1.23.1 or higher)
- curl
- Python 3.6+
- Python virtualenv package

4.1.10.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the Websocket Proxy:

- 4 GB of RAM
- 50 GB of free disk space

4.1.10.3 Setup

1. Change to the working directory by entering the following command:

   ```
   cd Projects/micronets/
   ```

   If you have not already created this directory, execute the following command:

   ```
   mkdir Projects/micronets/
   ```

   Next, change directories by entering the following command:

   ```
   cd Projects/micronets/
   ```

2. Download and install the cert generation scripts by executing the following commands:

   a. Download the script to generate the root certificates:

   ```
   curl -s -O https://raw.githubusercontent.com/cablelabs/micronets-ws-proxy/nccoe-build-3/bin/gen-root-cert
   ```

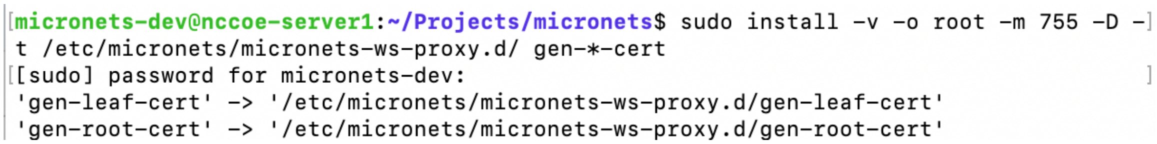
   b. Download the script to generate leaf certificates:

   ```
   curl -s -O https://raw.githubusercontent.com/cablelabs/micronets-ws-proxy/nccoe-build-3/bin/gen-leaf-cert
   ```

   c. Install both scripts by executing the following command:

   ```
   sudo install -v -o root -m 755 -D -t /etc/micronets/micronets-ws-proxy.d/
   gen-*-cert
   ```

   You should see output similar to the following:

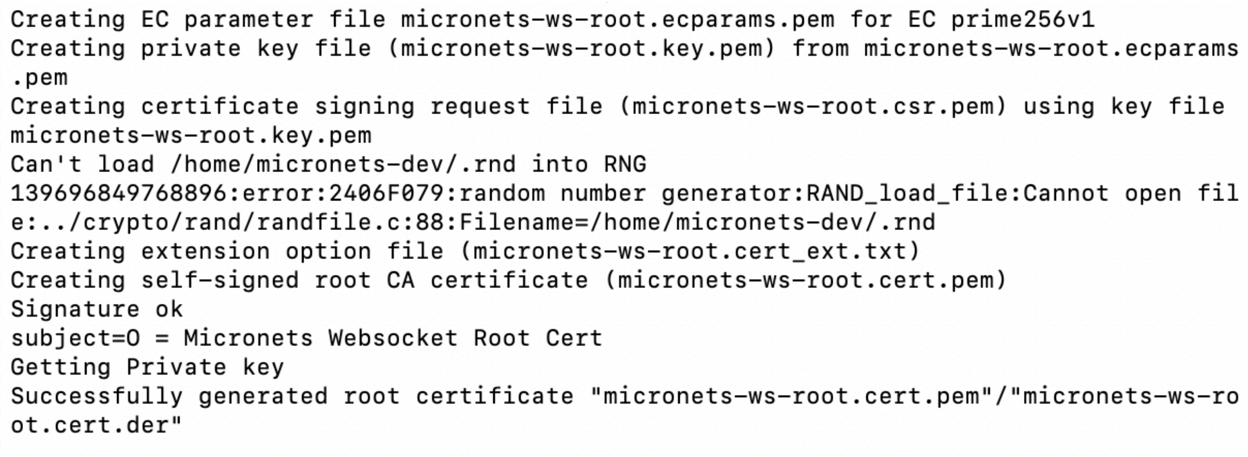
   

   ```
   [micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 -D -]
    t /etc/micronets/micronets-ws-proxy.d/ gen-*-cert
   [[sudo] password for micronets-dev:                                               ]
    'gen-leaf-cert' -> '/etc/micronets/micronets-ws-proxy.d/gen-leaf-cert'
    'gen-root-cert' -> '/etc/micronets/micronets-ws-proxy.d/gen-root-cert'
   ```

3. Create the root certificate for the Websocket Proxy:

   ```
   /etc/micronets/micronets-ws-proxy.d/gen-root-cert --cert-basename micronets-ws-
   root \
       --subject-org-name "Micronets Websocket Root Cert" \
       --expiration-in-days 3650
   ```

   You should see output similar to the following:

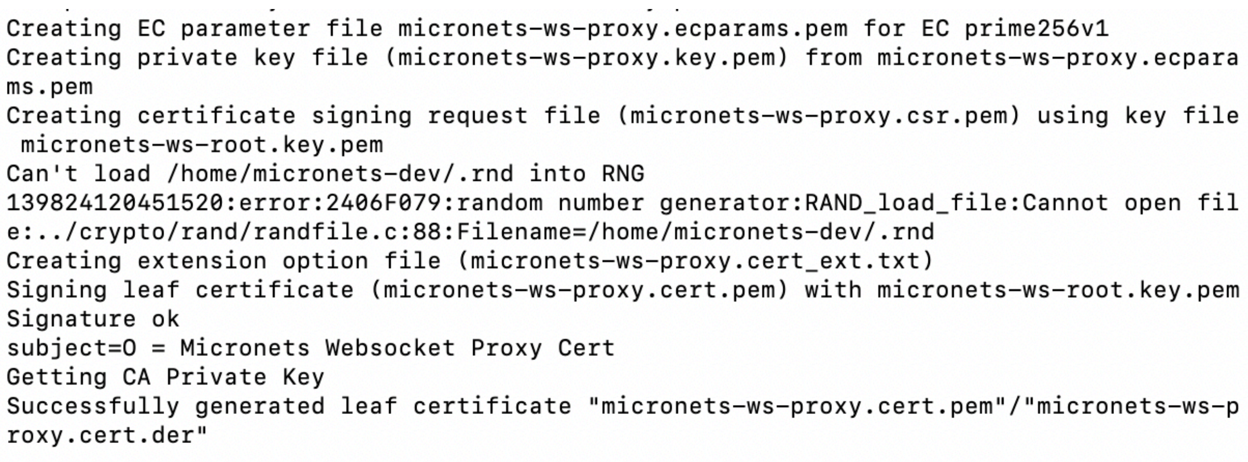


```
Creating EC parameter file micronets-ws-root.ecparams.pem for EC prime256v1
Creating private key file (micronets-ws-root.key.pem) from micronets-ws-root.ecparams
.pem
Creating certificate signing request file (micronets-ws-root.csr.pem) using key file
micronets-ws-root.key.pem
Can't load /home/micronets-dev/.rnd into RNG
139696849768896:error:2406F079:random number generator:RAND_load_file:Cannot open fil
e:../crypto/rand/randfile.c:88:Filename=/home/micronets-dev/.rnd
Creating extension option file (micronets-ws-root.cert_ext.txt)
Creating self-signed root CA certificate (micronets-ws-root.cert.pem)
Signature ok
subject=O = Micronets Websocket Root Cert
Getting Private key
Successfully generated root certificate "micronets-ws-root.cert.pem"/"micronets-ws-ro
ot.cert.der"
```

4. Create the Websocket Proxy's server certificate and private key by entering the following command (Note: This certificate and key host the Websocket Proxy server):

```
/etc/micronets/micronets-ws-proxy.d/gen-leaf-cert --cert-basename micronets-ws-
proxy \
  --subject-org-name "Micronets Websocket Proxy Cert" \
  --expiration-in-days 3650 \
  --ca-certfile micronets-ws-root.cert.pem \
  --ca-keyfile micronets-ws-root.key.pem
```

You should see output similar to the following:



```
Creating EC parameter file micronets-ws-proxy.ecparams.pem for EC prime256v1
Creating private key file (micronets-ws-proxy.key.pem) from micronets-ws-proxy.ecpara
ms.pem
Creating certificate signing request file (micronets-ws-proxy.csr.pem) using key file
 micronets-ws-root.key.pem
Can't load /home/micronets-dev/.rnd into RNG
139824120451520:error:2406F079:random number generator:RAND_load_file:Cannot open fil
e:../crypto/rand/randfile.c:88:Filename=/home/micronets-dev/.rnd
Creating extension option file (micronets-ws-proxy.cert_ext.txt)
Signing leaf certificate (micronets-ws-proxy.cert.pem) with micronets-ws-root.key.pem
Signature ok
subject=O = Micronets Websocket Proxy Cert
Getting CA Private Key
Successfully generated leaf certificate "micronets-ws-proxy.cert.pem"/"micronets-ws-p
roxy.cert.der"
```

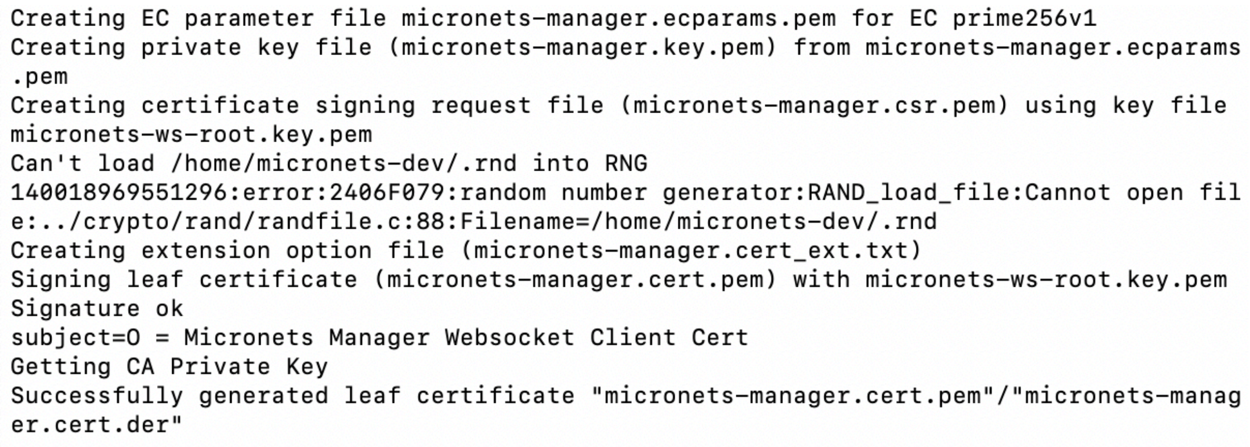
5. Combine the private key and certificate into one file by entering the following command:

```
cat micronets-ws-proxy.cert.pem micronets-ws-proxy.key.pem \
    > micronets-ws-proxy.pkeycert.pem
```

6. Generate the client certificate and key to be used by the Micronets Manager to connect to the Websocket Proxy (Note: These files will enable the Micronets Manager to connect to the proxy):

```
 /etc/micronets/micronets-ws-proxy.d/gen-leaf-cert --cert-basename micronets-manager \
     --subject-org-name "Micronets Manager Websocket Client Cert" \
     --expiration-in-days 3650 \
     --ca-certfile micronets-ws-root.cert.pem \
     --ca-keyfile micronets-ws-root.key.pem
```

You should see output similar to the following:



```
Creating EC parameter file micronets-manager.ecparams.pem for EC prime256v1
Creating private key file (micronets-manager.key.pem) from micronets-manager.ecparams
.pem
Creating certificate signing request file (micronets-manager.csr.pem) using key file
micronets-ws-root.key.pem
Can't load /home/micronets-dev/.rnd into RNG
140018969551296:error:2406F079:random number generator:RAND_load_file:Cannot open fil
e:../crypto/rand/randfile.c:88:Filename=/home/micronets-dev/.rnd
Creating extension option file (micronets-manager.cert_ext.txt)
Signing leaf certificate (micronets-manager.cert.pem) with micronets-ws-root.key.pem
Signature ok
subject=O = Micronets Manager Websocket Client Cert
Getting CA Private Key
Successfully generated leaf certificate "micronets-manager.cert.pem"/"micronets-manag
er.cert.der"
```

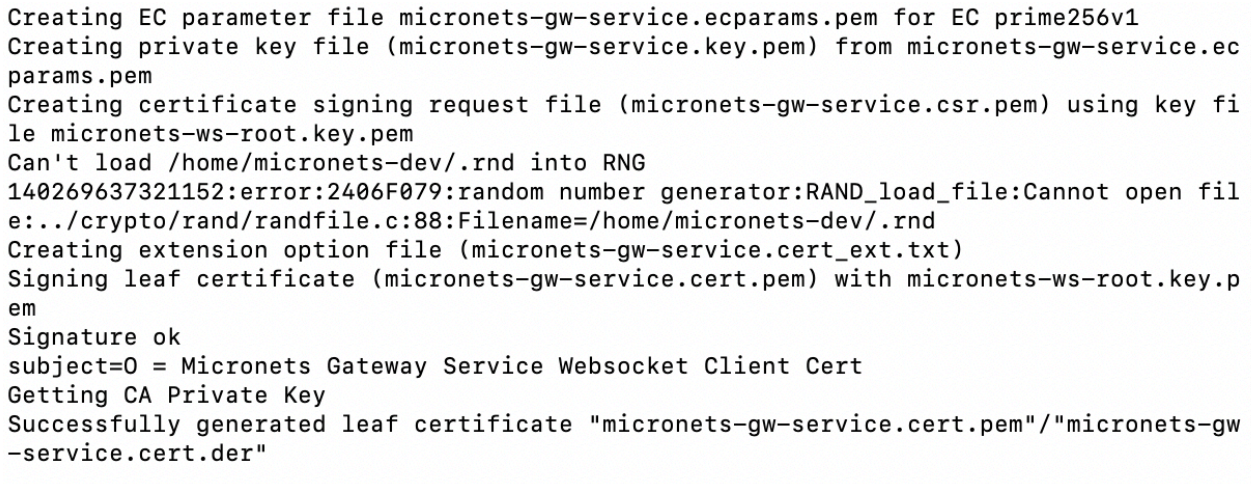
7. Combine the private key and certificate into one file by entering the following command:

```
 cat micronets-manager.cert.pem micronets-manager.key.pem \
     > micronets-manager.pkeycert.pem
```

8. Generate the certificate and key to be used by the Micronets Gateway to connect to the Websocket Proxy (Note: These files will enable the Micronets Gateway to connect to the proxy):

```
 /etc/micronets/micronets-ws-proxy.d/gen-leaf-cert --cert-basename micronets-gw-service \
     --subject-org-name "Micronets Gateway Service Websocket Client Cert" \
     --expiration-in-days 3650 \
     --ca-certfile micronets-ws-root.cert.pem \
     --ca-keyfile micronets-ws-root.key.pem
```

You should see output similar to the following:



```
Creating EC parameter file micronets-gw-service.ecparams.pem for EC prime256v1
Creating private key file (micronets-gw-service.key.pem) from micronets-gw-service.ec
params.pem
Creating certificate signing request file (micronets-gw-service.csr.pem) using key fi
le micronets-ws-root.key.pem
Can't load /home/micronets-dev/.rnd into RNG
140269637321152:error:2406F079:random number generator:RAND_load_file:Cannot open fil
e:../crypto/rand/randfile.c:88:Filename=/home/micronets-dev/.rnd
Creating extension option file (micronets-gw-service.cert_ext.txt)
Signing leaf certificate (micronets-gw-service.cert.pem) with micronets-ws-root.key.p
em
Signature ok
subject=O = Micronets Gateway Service Websocket Client Cert
Getting CA Private Key
Successfully generated leaf certificate "micronets-gw-service.cert.pem"/"micronets-gw
-service.cert.der"
```

9. Combine the private key and certificate into one file by entering the following command:

```
cat micronets-gw-service.cert.pem micronets-gw-service.key.pem \
    > micronets-gw-service.pkeycert.pem
```

10. Download and install the management script by entering the following commands:

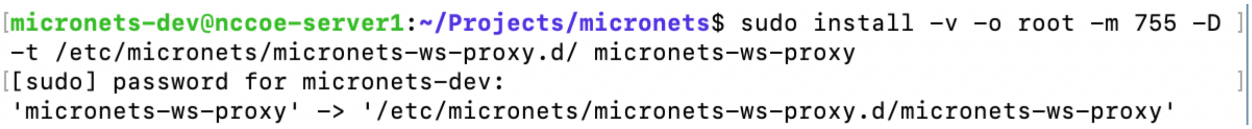
    a. Download the micronets-ws-proxy script:

    ```
    curl -s -O https://raw.githubusercontent.com/cablelabs/micronets-ws-proxy/nccoe-build-3/bin/micronets-ws-proxy
    ```

    b. Install the script to the appropriate directory:

    ```
    sudo install -v -o root -m 755 -D -t /etc/micronets/micronets-ws-proxy.d/
    micronets-ws-proxy
    ```

    You should see output similar to the following:

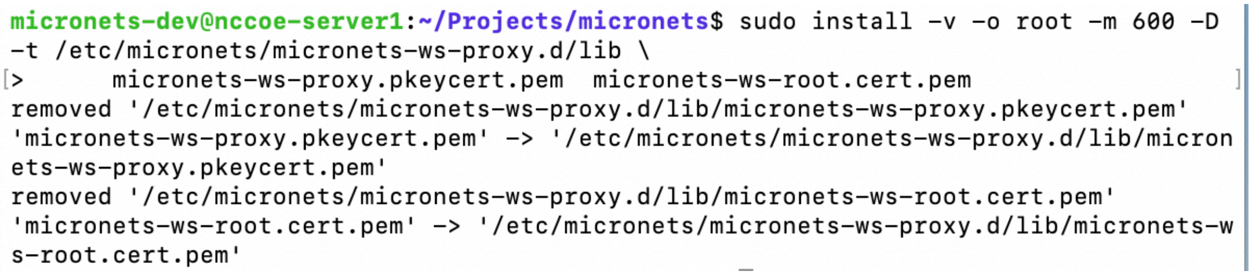


```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 -D ]
 -t /etc/micronets/micronets-ws-proxy.d/ micronets-ws-proxy
[[sudo] password for micronets-dev:                                                  ]
 'micronets-ws-proxy' -> '/etc/micronets/micronets-ws-proxy.d/micronets-ws-proxy'
```

11. Copy the Websocket Proxy server cert and key for use by the Websocket Proxy docker container:

```
sudo install -v -o root -m 600 -D -t /etc/micronets/micronets-ws-proxy.d/lib \

  micronets-ws-proxy.pkeycert.pem  micronets-ws-root.cert.pem
```

You should see output similar to the following:

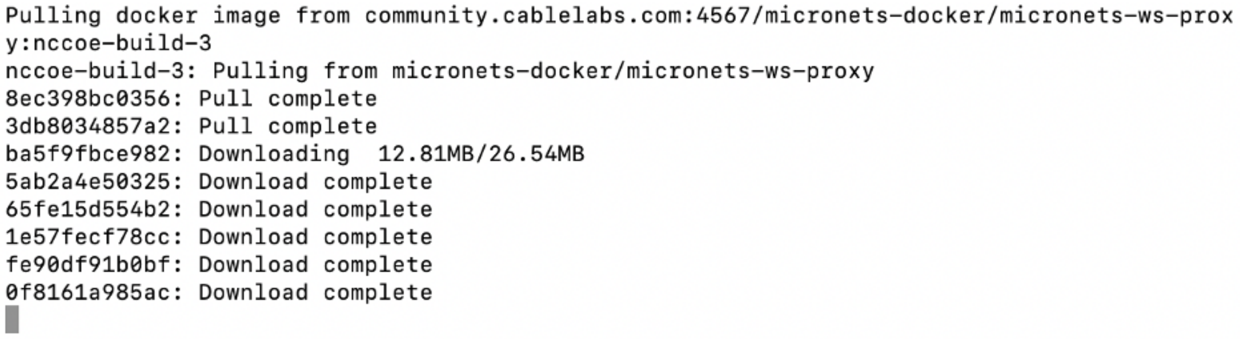


```
micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 600 -D
-t /etc/micronets/micronets-ws-proxy.d/lib \
[>      micronets-ws-proxy.pkeycert.pem  micronets-ws-root.cert.pem                  ]
removed '/etc/micronets/micronets-ws-proxy.d/lib/micronets-ws-proxy.pkeycert.pem'
'micronets-ws-proxy.pkeycert.pem' -> '/etc/micronets/micronets-ws-proxy.d/lib/micron
ets-ws-proxy.pkeycert.pem'
removed '/etc/micronets/micronets-ws-proxy.d/lib/micronets-ws-root.cert.pem'
'micronets-ws-root.cert.pem' -> '/etc/micronets/micronets-ws-proxy.d/lib/micronets-w
s-root.cert.pem'
```

12. Download the Micronets Websocket Proxy docker image (Note: If you cannot connect to the docker service, use sudo usermod -aG docker to add the user account to the docker group):

```
/etc/micronets/micronets-ws-proxy.d/micronets-ws-proxy docker-pull
```

You should see output similar to the following:

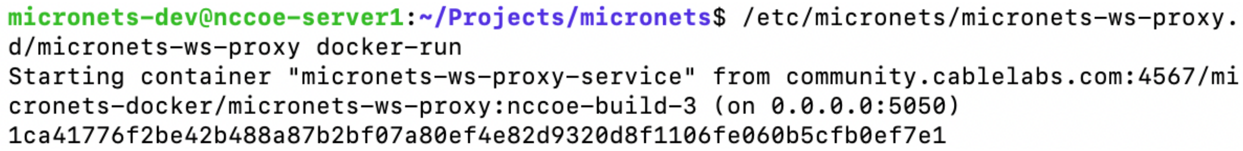


```
Pulling docker image from community.cablelabs.com:4567/micronets-docker/micronets-ws-prox
y:nccoe-build-3
nccoe-build-3: Pulling from micronets-docker/micronets-ws-proxy
8ec398bc0356: Pull complete
3db8034857a2: Pull complete
ba5f9fbce982: Downloading  12.81MB/26.54MB
5ab2a4e50325: Download complete
65fe15d554b2: Download complete
1e57fecf78cc: Download complete
fe90df91b0bf: Download complete
0f8161a985ac: Download complete
```

13. Start the Websocket Proxy:

```
/etc/micronets/micronets-ws-proxy.d/micronets-ws-proxy docker-run
```

You should see output similar to the following:

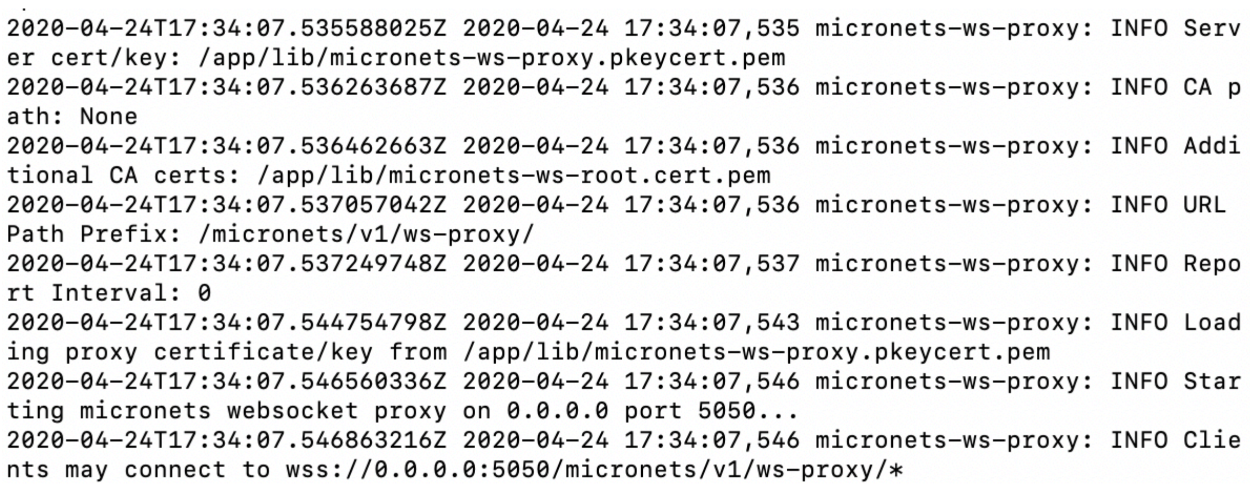


```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/micronets-ws-proxy.
d/micronets-ws-proxy docker-run
Starting container "micronets-ws-proxy-service" from community.cablelabs.com:4567/mi
cronets-docker/micronets-ws-proxy:nccoe-build-3 (on 0.0.0.0:5050)
1ca41776f2be42b488a87b2bf07a80ef4e82d9320d8f1106fe060b5cfb0ef7e1
```

14. Verify that the Websocket Proxy is running:

```
/etc/micronets/micronets-ws-proxy.d/micronets-ws-proxy docker-logs
```

You should see output similar to the following:



```
2020-04-24T17:34:07.535588025Z 2020-04-24 17:34:07,535 micronets-ws-proxy: INFO Serv
er cert/key: /app/lib/micronets-ws-proxy.pkeycert.pem
2020-04-24T17:34:07.536263687Z 2020-04-24 17:34:07,536 micronets-ws-proxy: INFO CA p
ath: None
2020-04-24T17:34:07.536462663Z 2020-04-24 17:34:07,536 micronets-ws-proxy: INFO Addi
tional CA certs: /app/lib/micronets-ws-root.cert.pem
2020-04-24T17:34:07.537057042Z 2020-04-24 17:34:07,536 micronets-ws-proxy: INFO URL
Path Prefix: /micronets/v1/ws-proxy/
2020-04-24T17:34:07.537249748Z 2020-04-24 17:34:07,537 micronets-ws-proxy: INFO Repo
rt Interval: 0
2020-04-24T17:34:07.544754798Z 2020-04-24 17:34:07,543 micronets-ws-proxy: INFO Load
ing proxy certificate/key from /app/lib/micronets-ws-proxy.pkeycert.pem
2020-04-24T17:34:07.546560336Z 2020-04-24 17:34:07,546 micronets-ws-proxy: INFO Star
ting micronets websocket proxy on 0.0.0.0 port 5050...
2020-04-24T17:34:07.546863216Z 2020-04-24 17:34:07,546 micronets-ws-proxy: INFO Clie
nts may connect to wss://0.0.0.0:5050/micronets/v1/ws-proxy/*
```

15. Verify the Websocket Proxy credentials by executing the following steps:

    a. Download the Websocket test client script:

    ```
    curl -O https://raw.githubusercontent.com/cablelabs/micronets-ws-proxy/nccoe-build-3/bin/websocket-test-client.py
    ```

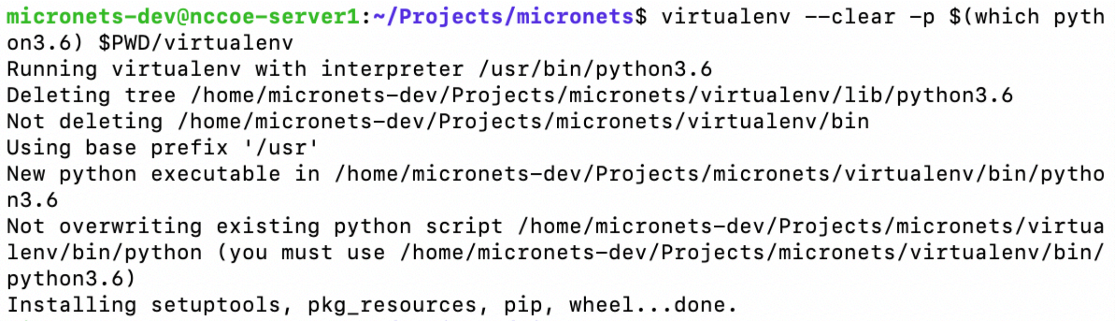
    b. Download the requirements text file:

    ```
    curl -O https://raw.githubusercontent.com/cablelabs/micronets-ws-proxy/nccoe-build-3/requirements.txt
    ```

c. Clear out the nonroot installation of virtualenv, and set the Python interpreter to use Python 3.6 for the script installation:

```
virtualenv --clear -p $(which python3.6) $PWD/virtualenv
```

You should see output similar to the following:

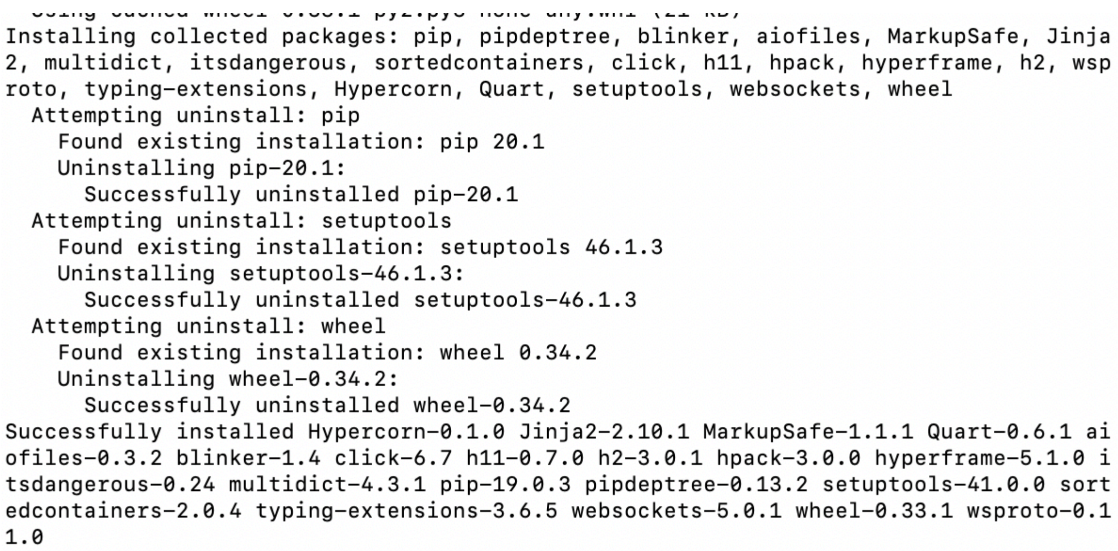


```
micronets-dev@nccoe-server1:~/Projects/micronets$ virtualenv --clear -p $(which pyth
on3.6) $PWD/virtualenv
Running virtualenv with interpreter /usr/bin/python3.6
Deleting tree /home/micronets-dev/Projects/micronets/virtualenv/lib/python3.6
Not deleting /home/micronets-dev/Projects/micronets/virtualenv/bin
Using base prefix '/usr'
New python executable in /home/micronets-dev/Projects/micronets/virtualenv/bin/pytho
n3.6
Not overwriting existing python script /home/micronets-dev/Projects/micronets/virtua
lenv/bin/python (you must use /home/micronets-dev/Projects/micronets/virtualenv/bin/
python3.6)
Installing setuptools, pkg_resources, pip, wheel...done.
```

d. Install virtualenv and pass the requirements text file:

```
./virtualenv/bin/pip install -r requirements.txt
```

You should see output similar to the following:

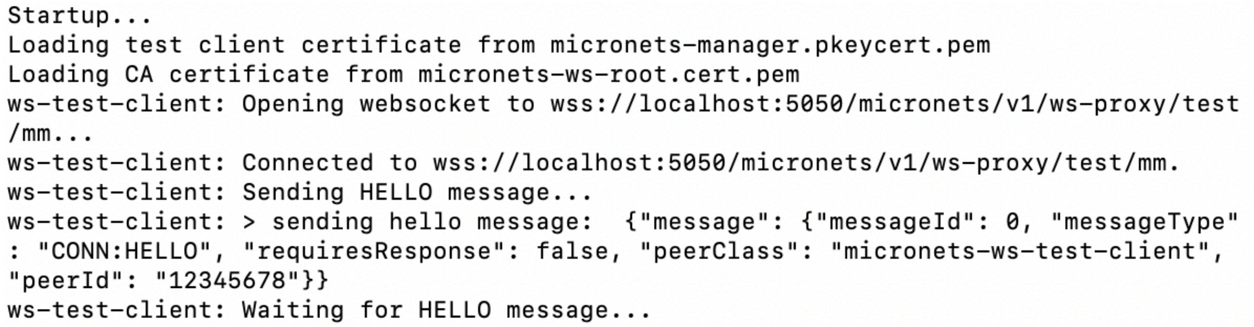


```
  Using cached wheel-0.33.1-py2.py3-none-any.whl (21 kB)
Installing collected packages: pip, pipdeptree, blinker, aiofiles, MarkupSafe, Jinja
2, multidict, itsdangerous, sortedcontainers, click, h11, hpack, hyperframe, h2, wsp
roto, typing-extensions, Hypercorn, Quart, setuptools, websockets, wheel
  Attempting uninstall: pip
    Found existing installation: pip 20.1
    Uninstalling pip-20.1:
      Successfully uninstalled pip-20.1
  Attempting uninstall: setuptools
    Found existing installation: setuptools 46.1.3
    Uninstalling setuptools-46.1.3:
      Successfully uninstalled setuptools-46.1.3
  Attempting uninstall: wheel
    Found existing installation: wheel 0.34.2
    Uninstalling wheel-0.34.2:
      Successfully uninstalled wheel-0.34.2
Successfully installed Hypercorn-0.1.0 Jinja2-2.10.1 MarkupSafe-1.1.1 Quart-0.6.1 ai
ofiles-0.3.2 blinker-1.4 click-6.7 h11-0.7.0 h2-3.0.1 hpack-3.0.0 hyperframe-5.1.0 i
tsdangerous-0.24 multidict-4.3.1 pip-19.0.3 pipdeptree-0.13.2 setuptools-41.0.0 sort
edcontainers-2.0.4 typing-extensions-3.6.5 websockets-5.0.1 wheel-0.33.1 wsproto-0.1
1.0
```

e. Run the Websocket test client script:

```
./virtualenv/bin/python websocket-test-client.py \
    --client-cert micronets-manager.pkeycert.pem \
    --ca-cert micronets-ws-root.cert.pem  \
wss://localhost:5050/micronets/v1/ws-proxy/test/mm
```

You should see output similar to the following:

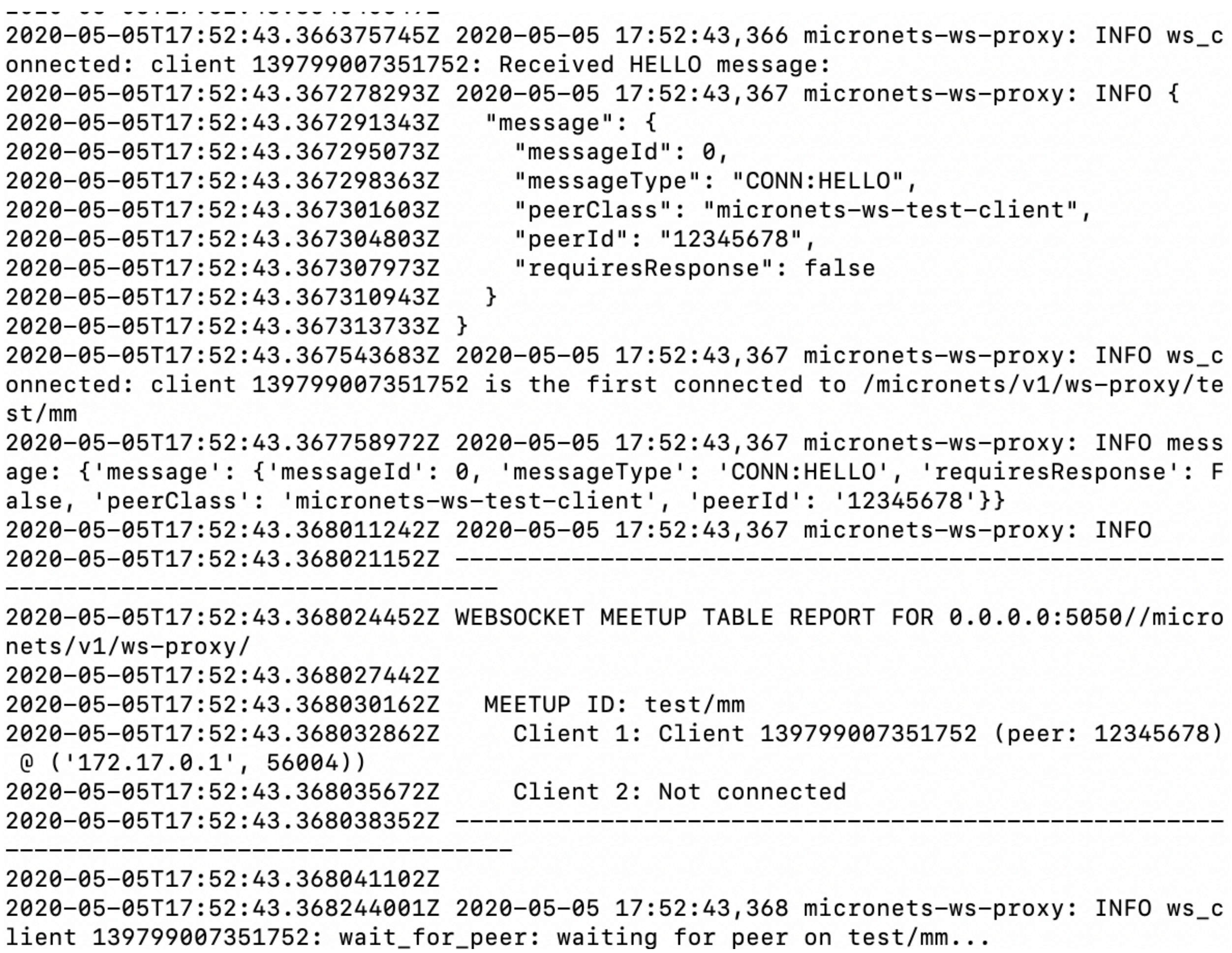


```
Startup...
Loading test client certificate from micronets-manager.pkeycert.pem
Loading CA certificate from micronets-ws-root.cert.pem
ws-test-client: Opening websocket to wss://localhost:5050/micronets/v1/ws-proxy/test
/mm...
ws-test-client: Connected to wss://localhost:5050/micronets/v1/ws-proxy/test/mm.
ws-test-client: Sending HELLO message...
ws-test-client: > sending hello message:  {"message": {"messageId": 0, "messageType"
: "CONN:HELLO", "requiresResponse": false, "peerClass": "micronets-ws-test-client",
"peerId": "12345678"}}
ws-test-client: Waiting for HELLO message...
```

f. Verify communication from the test client to the Websocket Proxy by checking the logs:

```
/etc/micronets/micronets-ws-proxy.d/micronets-ws-proxy docker-logs
```

You should see output similar to the following:



```
2020-05-05T17:52:43.366375745Z 2020-05-05 17:52:43,366 micronets-ws-proxy: INFO ws_c
onnected: client 139799007351752: Received HELLO message:
2020-05-05T17:52:43.367278293Z 2020-05-05 17:52:43,367 micronets-ws-proxy: INFO {
2020-05-05T17:52:43.367291343Z   "message": {
2020-05-05T17:52:43.367295073Z     "messageId": 0,
2020-05-05T17:52:43.367298363Z     "messageType": "CONN:HELLO",
2020-05-05T17:52:43.367301603Z     "peerClass": "micronets-ws-test-client",
2020-05-05T17:52:43.367304803Z     "peerId": "12345678",
2020-05-05T17:52:43.367307973Z     "requiresResponse": false
2020-05-05T17:52:43.367310943Z   }
2020-05-05T17:52:43.367313733Z }
2020-05-05T17:52:43.367543683Z 2020-05-05 17:52:43,367 micronets-ws-proxy: INFO ws_c
onnected: client 139799007351752 is the first connected to /micronets/v1/ws-proxy/te
st/mm
2020-05-05T17:52:43.367758972Z 2020-05-05 17:52:43,367 micronets-ws-proxy: INFO mess
age: {'message': {'messageId': 0, 'messageType': 'CONN:HELLO', 'requiresResponse': F
alse, 'peerClass': 'micronets-ws-test-client', 'peerId': '12345678'}}
2020-05-05T17:52:43.368011242Z 2020-05-05 17:52:43,367 micronets-ws-proxy: INFO
2020-05-05T17:52:43.368021152Z -----------------------------------------------------
----------------------------------
2020-05-05T17:52:43.368024452Z WEBSOCKET MEETUP TABLE REPORT FOR 0.0.0.0:5050//micro
nets/v1/ws-proxy/
2020-05-05T17:52:43.368027442Z
2020-05-05T17:52:43.368030162Z   MEETUP ID: test/mm
2020-05-05T17:52:43.368032862Z     Client 1: Client 139799007351752 (peer: 12345678)
 @ ('172.17.0.1', 56004))
2020-05-05T17:52:43.368035672Z     Client 2: Not connected
2020-05-05T17:52:43.368038352Z -----------------------------------------------------
-----------------------------------
2020-05-05T17:52:43.368041102Z
2020-05-05T17:52:43.368244001Z 2020-05-05 17:52:43,368 micronets-ws-proxy: INFO ws_c
lient 139799007351752: wait_for_peer: waiting for peer on test/mm...
```

16. Save the *micronets-manager.pkeycert.pem, micronets-gw-service.pkeycert.pem,* and *micronets-ws-root.cert.pem* files for configuring the Micronets Manager and Micronets Gateway components.

4.1.11 Micronets iPhone Application for Device Onboarding

This section describes the CableLabs Micronets iPhone application, which is a mobile application used for onboarding DPP-capable devices. This implementation leverages the latest CableLabs Micronets iPhone application Git release. This documentation describes setting up your own Micronets iPhone application.

4.1.11.1 Micronets iPhone Application Overview

The Micronets iPhone application is responsible for sending onboarding requests and related elements to the MSO portal when the user initiates the onboarding process on the Micronets Proto-Pi device and scans the QR code. If building with an Android phone, follow the documentation provided here: https://github.com/cablelabs/micronets-mobile/blob/nccoe-build-3/README.md#android

4.1.11.2 Configuration Overview

The following subsections document the software and network configurations for the Micronets iPhone application.

4.1.11.2.1 Network Configuration

The mobile phone on which the Micronets application is being installed should have internet access via either the cellular network or Wi-Fi.

4.1.11.2.2 Software Configuration

The following software is required to install, configure, and operate the Micronets iPhone application:

- macOS (minimum version 10.13; High Sierra)
- Apple iOS Developer license
- Node (minimum version 8)
- Cordova (version 8.0.0; problems with version 9)
- Xcode (minimum version 9.2)
- ImageMagick
- Brew

4.1.11.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the Micronets iPhone application:

- Apple computing system (laptop or desktop)
- Apple iPhone (any model compatible with iOS 10.3 and above)

4.1.11.3 Setup

4.1.11.3.1 Install Dependencies

1. Install Node by entering the following command in the terminal:

```
brew install node
```

2. Install ImageMagick by entering the following command in the terminal:

   ```
   brew install imagemagick
   ```

3. Install Cordova version 8.0.0 by entering the following command:

   ```
   sudo npm install -g cordova@8.0.0
   ```

4. Install ios-deploy, which Cordova uses to cable-load the application, by entering the following command:

   ```
   sudo npm install -g --unsafe-perm=true ios-deploy
   ```

   Note: The `unsafe-perm` flag is required on macOS versions El Capitan and higher.

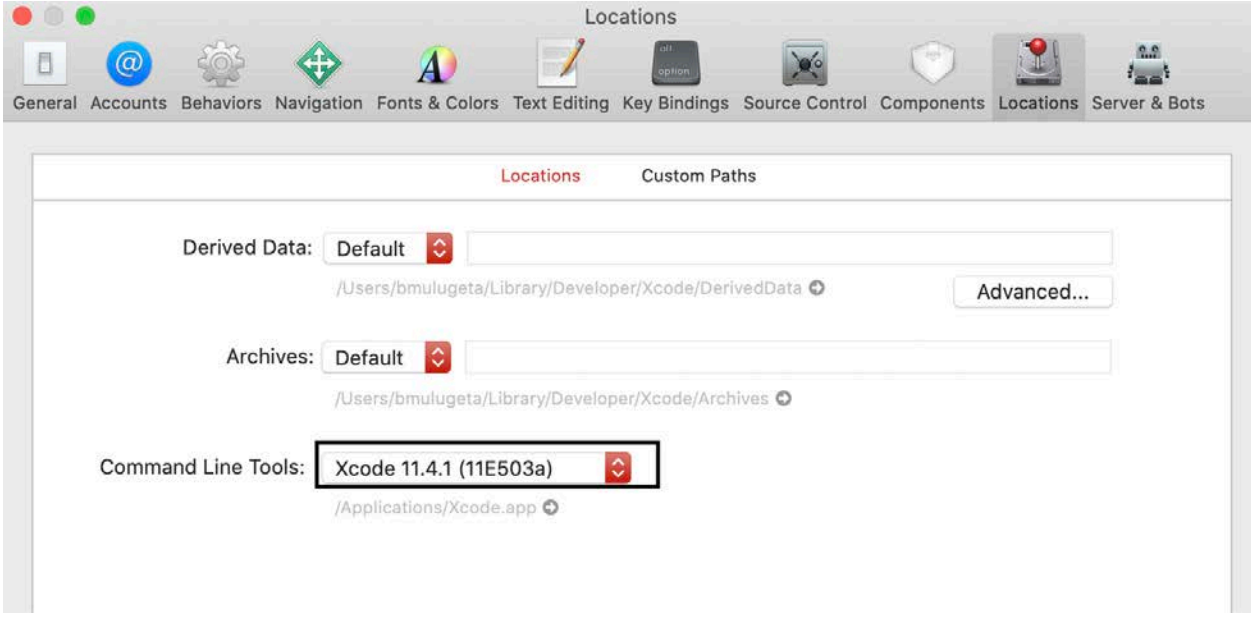
   If you run into an EACCES: permission denied error, attempt the following fixes:

   ```
   sudo chown -R $USER:$GROUP ~/.npm

   sudo chown -R $USER:$GROUP ~/.config
   ```

5. Open Xcode, and add Xcode to your command-line tools:

   Preferences > Location > Command Line Tools

   Select your Xcode version as seen in screenshot below:



Derived Data: Default
/Users/bmulugeta/Library/Developer/Xcode/DerivedData

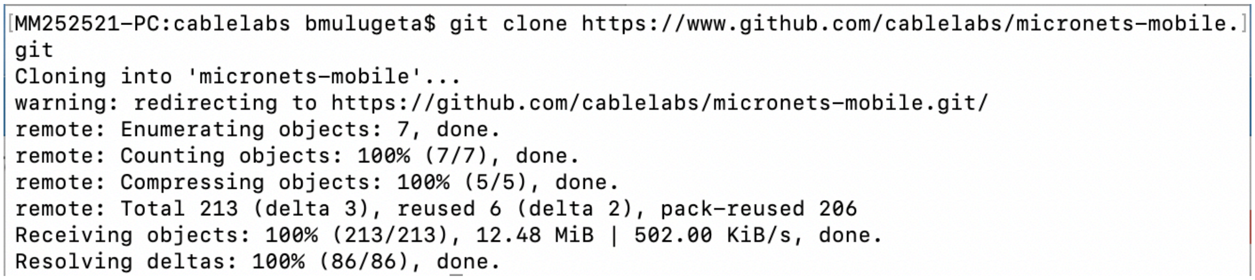
Archives: Default
/Users/bmulugeta/Library/Developer/Xcode/Archives

Command Line Tools: Xcode 11.4.1 (11E503a)
/Applications/Xcode.app

4.1.11.3.2 Build Micronets iPhone Application

1. Check out the repo that contains the Micronets mobile application build by entering the following command:

   ```
   git clone https://www.github.com/cablelabs/micronets-mobile.git
   ```



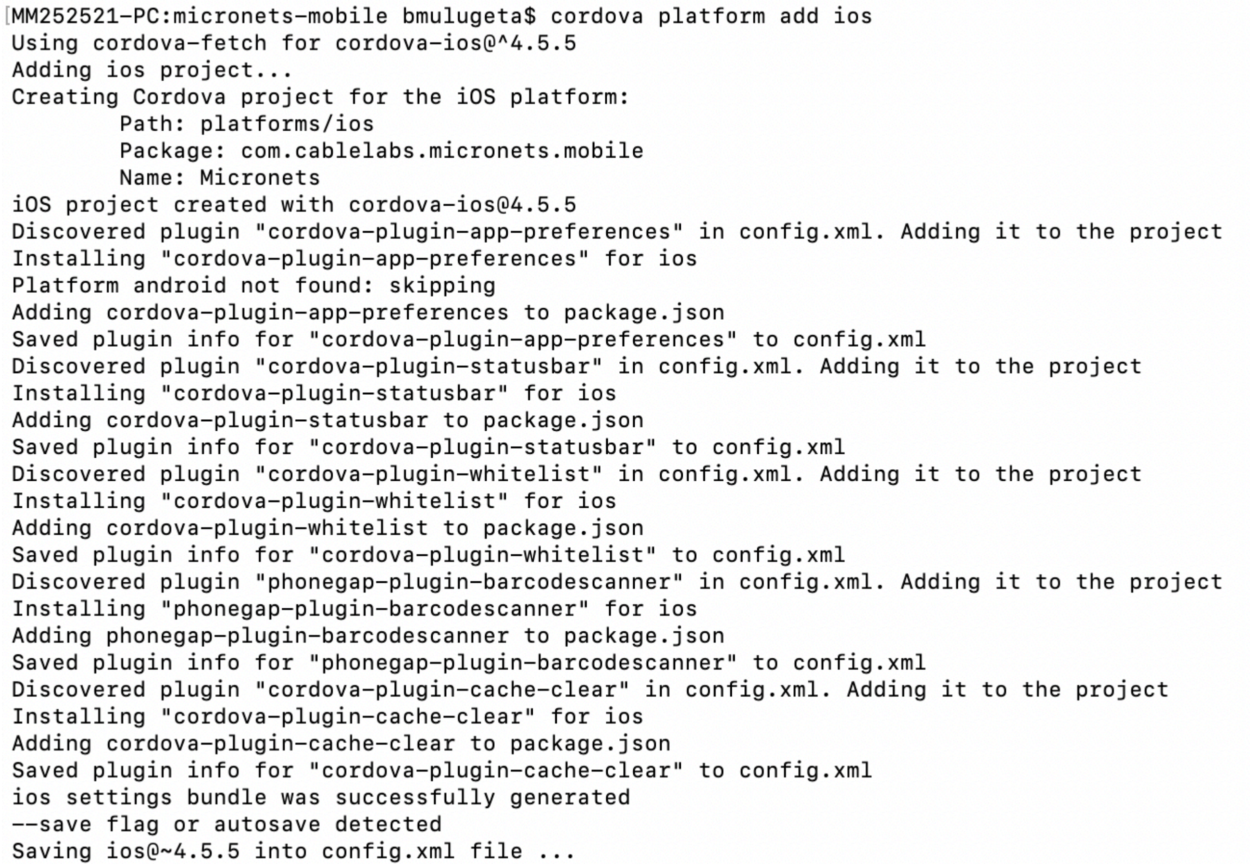
```
MM252521-PC:cablelabs bmulugeta$ git clone https://www.github.com/cablelabs/micronets-mobile.
git
Cloning into 'micronets-mobile'...
warning: redirecting to https://github.com/cablelabs/micronets-mobile.git/
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 213 (delta 3), reused 6 (delta 2), pack-reused 206
Receiving objects: 100% (213/213), 12.48 MiB | 502.00 KiB/s, done.
Resolving deltas: 100% (86/86), done.
```

2. Enter the Micronets mobile directory by entering the following command:

   ```
   cd micronets-mobile
   ```

3. Add the target platform by entering the following command:

   ```
   cordova platform add ios
   ```

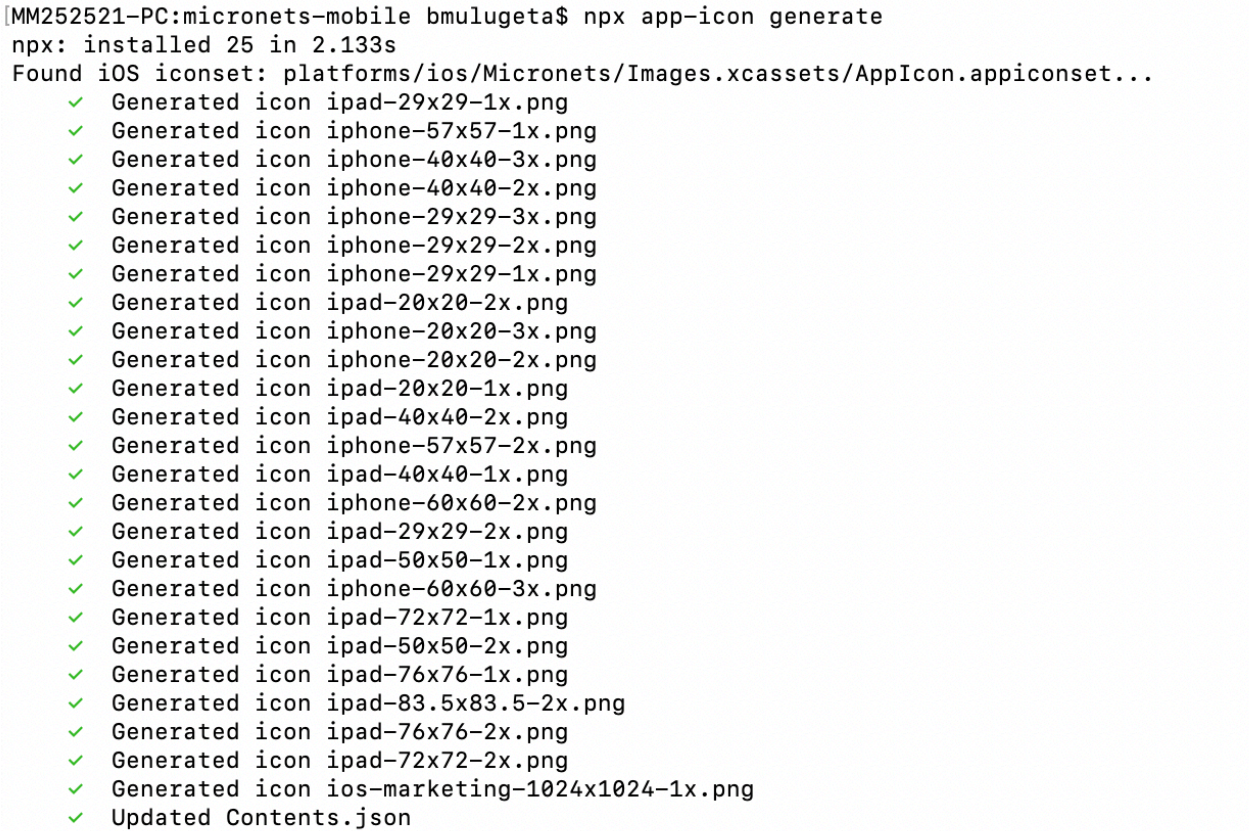


```
[MM252521-PC:micronets-mobile bmulugeta$ cordova platform add ios
 Using cordova-fetch for cordova-ios@^4.5.5
 Adding ios project...
 Creating Cordova project for the iOS platform:
         Path: platforms/ios
         Package: com.cablelabs.micronets.mobile
         Name: Micronets
 iOS project created with cordova-ios@4.5.5
 Discovered plugin "cordova-plugin-app-preferences" in config.xml. Adding it to the project
 Installing "cordova-plugin-app-preferences" for ios
 Platform android not found: skipping
 Adding cordova-plugin-app-preferences to package.json
 Saved plugin info for "cordova-plugin-app-preferences" to config.xml
 Discovered plugin "cordova-plugin-statusbar" in config.xml. Adding it to the project
 Installing "cordova-plugin-statusbar" for ios
 Adding cordova-plugin-statusbar to package.json
 Saved plugin info for "cordova-plugin-statusbar" to config.xml
 Discovered plugin "cordova-plugin-whitelist" in config.xml. Adding it to the project
 Installing "cordova-plugin-whitelist" for ios
 Adding cordova-plugin-whitelist to package.json
 Saved plugin info for "cordova-plugin-whitelist" to config.xml
 Discovered plugin "phonegap-plugin-barcodescanner" in config.xml. Adding it to the project
 Installing "phonegap-plugin-barcodescanner" for ios
 Adding phonegap-plugin-barcodescanner to package.json
 Saved plugin info for "phonegap-plugin-barcodescanner" to config.xml
 Discovered plugin "cordova-plugin-cache-clear" in config.xml. Adding it to the project
 Installing "cordova-plugin-cache-clear" for ios
 Adding cordova-plugin-cache-clear to package.json
 Saved plugin info for "cordova-plugin-cache-clear" to config.xml
 ios settings bundle was successfully generated
 --save flag or autosave detected
 Saving ios@~4.5.5 into config.xml file ...
```

4. Generate iOS icon set by entering the following command:

```
npx app-icon generate
```

You should see the following output:



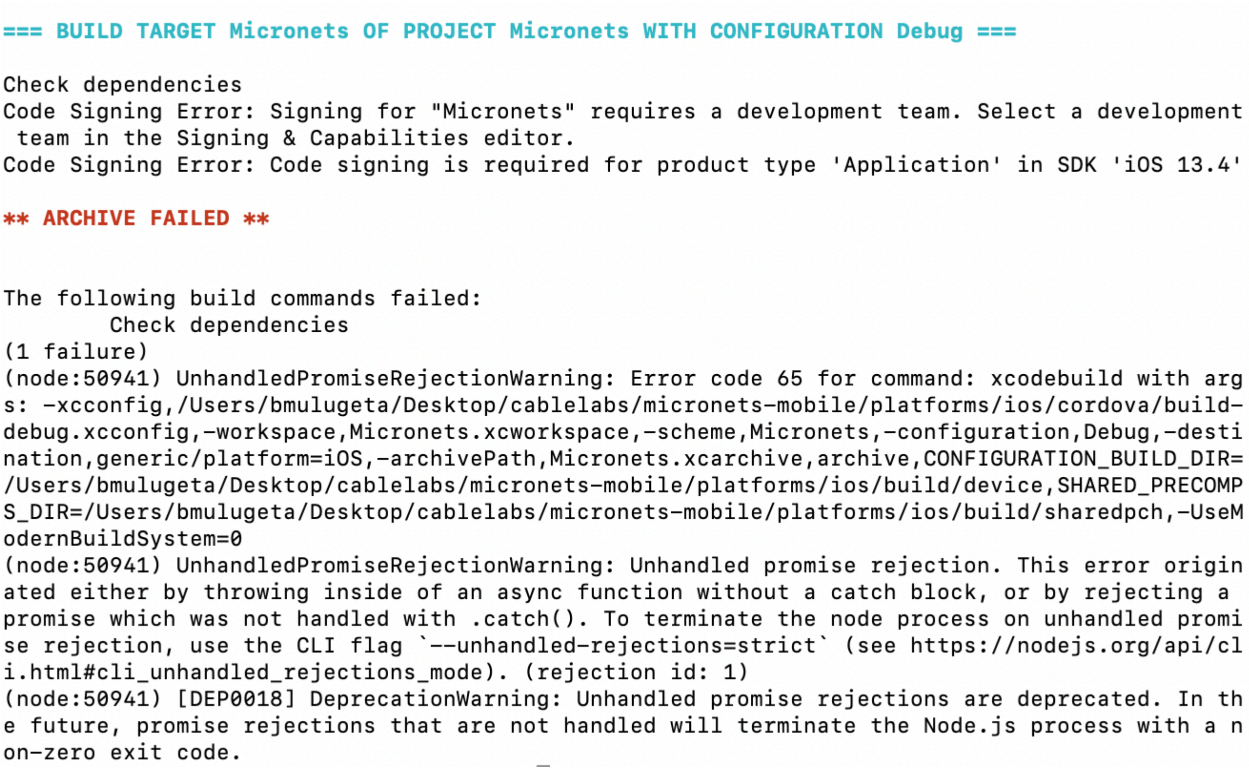
```
[MM252521-PC:micronets-mobile bmulugeta$ npx app-icon generate
 npx: installed 25 in 2.133s
 Found iOS iconset: platforms/ios/Micronets/Images.xcassets/AppIcon.appiconset...
     ✓  Generated icon ipad-29x29-1x.png
     ✓  Generated icon iphone-57x57-1x.png
     ✓  Generated icon iphone-40x40-3x.png
     ✓  Generated icon iphone-40x40-2x.png
     ✓  Generated icon iphone-29x29-3x.png
     ✓  Generated icon iphone-29x29-2x.png
     ✓  Generated icon iphone-29x29-1x.png
     ✓  Generated icon ipad-20x20-2x.png
     ✓  Generated icon iphone-20x20-3x.png
     ✓  Generated icon iphone-20x20-2x.png
     ✓  Generated icon ipad-20x20-1x.png
     ✓  Generated icon ipad-40x40-2x.png
     ✓  Generated icon iphone-57x57-2x.png
     ✓  Generated icon ipad-40x40-1x.png
     ✓  Generated icon iphone-60x60-2x.png
     ✓  Generated icon ipad-29x29-2x.png
     ✓  Generated icon ipad-50x50-1x.png
     ✓  Generated icon iphone-60x60-3x.png
     ✓  Generated icon ipad-72x72-1x.png
     ✓  Generated icon ipad-50x50-2x.png
     ✓  Generated icon ipad-76x76-1x.png
     ✓  Generated icon ipad-83.5x83.5-2x.png
     ✓  Generated icon ipad-76x76-2x.png
     ✓  Generated icon ipad-72x72-2x.png
     ✓  Generated icon ios-marketing-1024x1024-1x.png
     ✓  Updated Contents.json
```

5. Plug your iPhone into your computer, unlock your phone, and open to home screen. (You will need to allow developer use of the phone. You will be prompted.)

6. Run the following command to build the mobile application:

   ```
   cordova run ios --device --buildFlag='-UseModernBuildSystem=0'
   ```

   You should see output similar to the following:

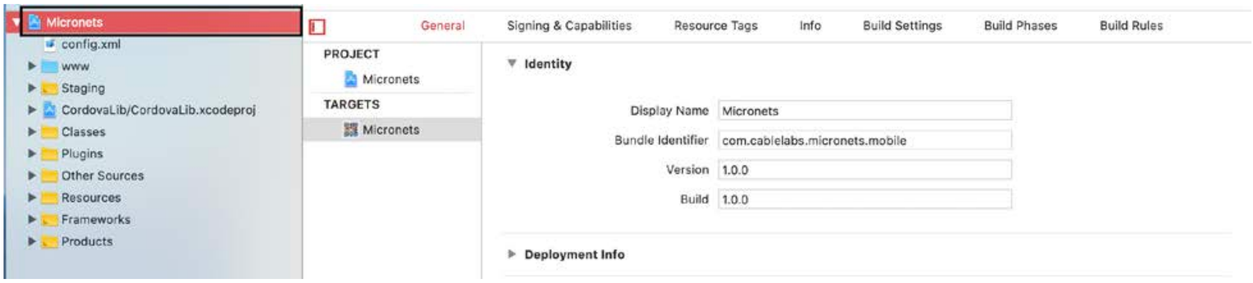


```
=== BUILD TARGET Micronets OF PROJECT Micronets WITH CONFIGURATION Debug ===

Check dependencies
Code Signing Error: Signing for "Micronets" requires a development team. Select a development
 team in the Signing & Capabilities editor.
Code Signing Error: Code signing is required for product type 'Application' in SDK 'iOS 13.4'

** ARCHIVE FAILED **


The following build commands failed:
        Check dependencies
(1 failure)
(node:50941) UnhandledPromiseRejectionWarning: Error code 65 for command: xcodebuild with arg
s: -xcconfig,/Users/bmulugeta/Desktop/cablelabs/micronets-mobile/platforms/ios/cordova/build-
debug.xcconfig,-workspace,Micronets.xcworkspace,-scheme,Micronets,-configuration,Debug,-desti
nation,generic/platform=iOS,-archivePath,Micronets.xcarchive,archive,CONFIGURATION_BUILD_DIR=
/Users/bmulugeta/Desktop/cablelabs/micronets-mobile/platforms/ios/build/device,SHARED_PRECOMP
S_DIR=/Users/bmulugeta/Desktop/cablelabs/micronets-mobile/platforms/ios/build/sharedpch,-UseM
odernBuildSystem=0
(node:50941) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error origin
ated either by throwing inside of an async function without a catch block, or by rejecting a
promise which was not handled with .catch(). To terminate the node process on unhandled promi
se rejection, use the CLI flag `--unhandled-rejections=strict` (see https://nodejs.org/api/cl
i.html#cli_unhandled_rejections_mode). (rejection id: 1)
(node:50941) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In th
e future, promise rejections that are not handled will terminate the Node.js process with a n
on-zero exit code.
```

Note: This initial attempt to build is expected to fail. It is necessary to open the project in Xcode and change some settings.

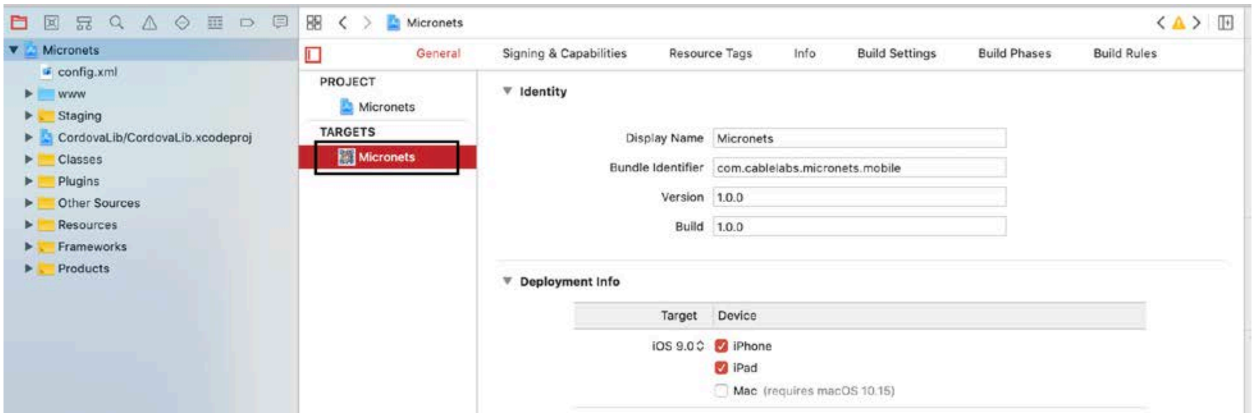
7. Open the project file *platforms/ios/Micronets.xcodeproj* in Xcode.

8. Click the Micronets icon in the navigator pane on the left. The properties pane should now be visible on the right:



Display Name: Micronets
Bundle Identifier: com.cablelabs.micronets.mobile
Version: 1.0.0
Build: 1.0.0

9. Select **Micronets** under **TARGETS:**



Identity

Display Name: Micronets

Bundle Identifier: com.cablelabs.micronets.mobile

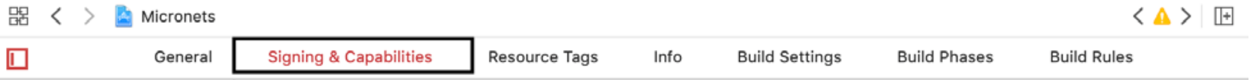
Version: 1.0.0

Build: 1.0.0

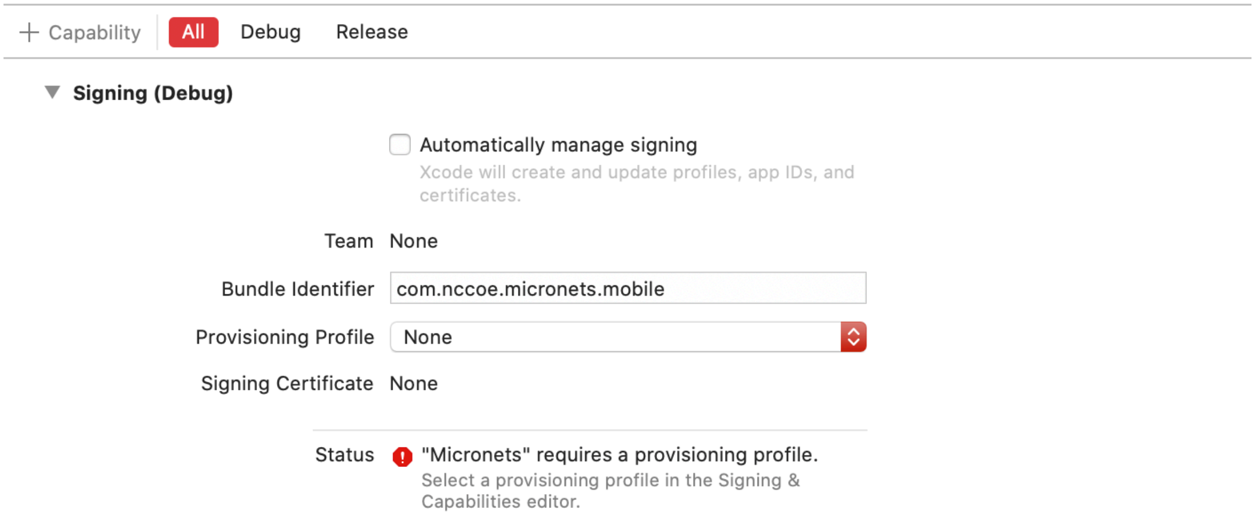
Deployment Info

Target	Device
iOS 9.0	☑ iPhone ☑ iPad □ Mac (requires macOS 10.15)

10. Select the **Signing & Capabilities** tab in the heading:



11. Ensure **Automatically manage signing** is checked:



Signing (Debug)

□ Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

Team: None

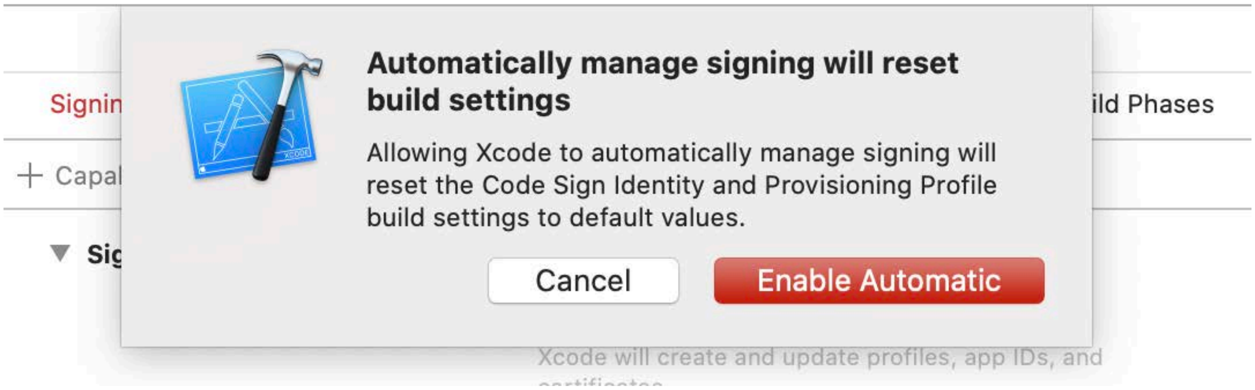
Bundle Identifier: com.nccoe.micronets.mobile

Provisioning Profile: None

Signing Certificate: None

Status: "Micronets" requires a provisioning profile.
Select a provisioning profile in the Signing & Capabilities editor.

You will see the following notification. Select **Enable Automatic:**

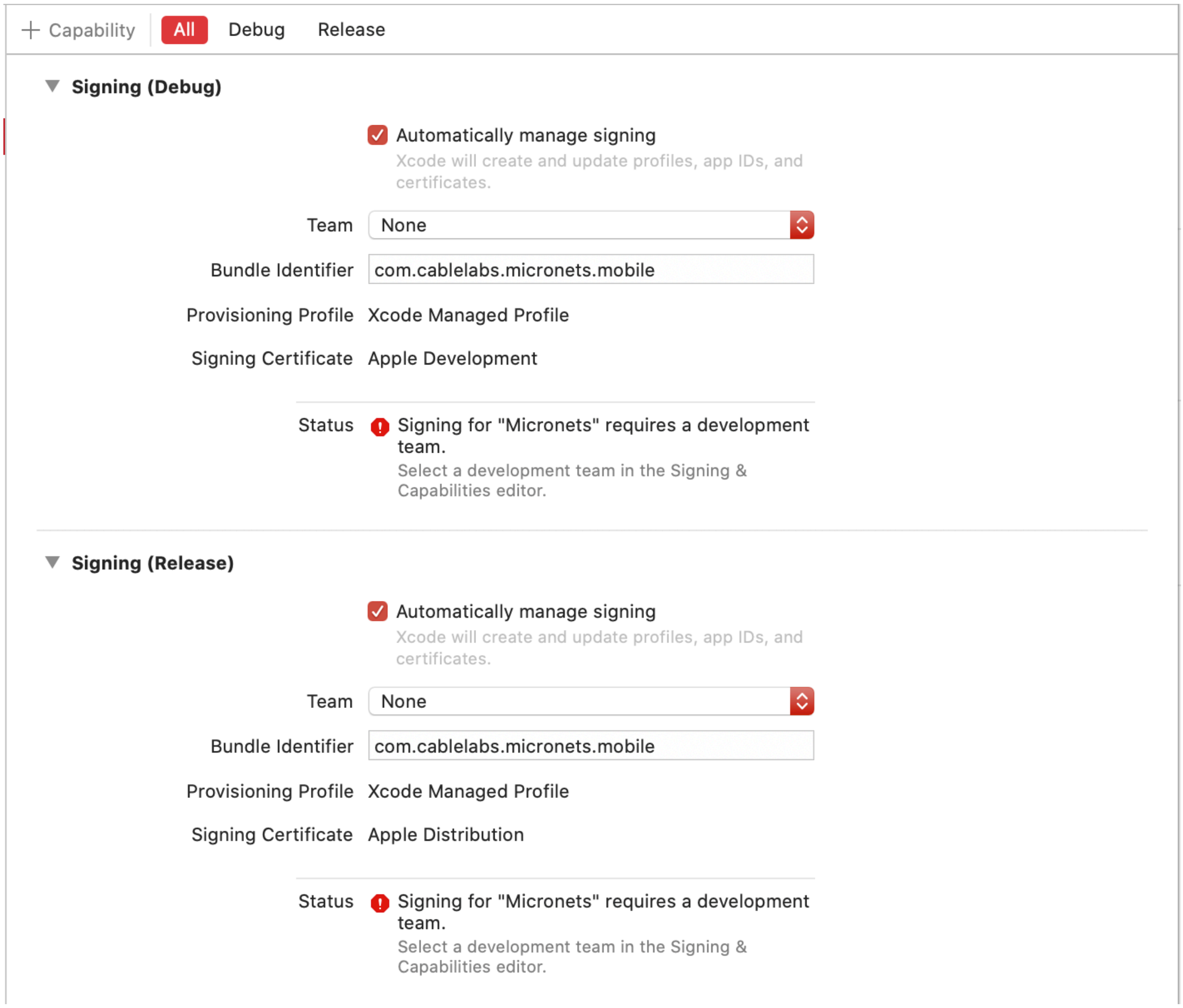


Automatically manage signing will reset build settings

Allowing Xcode to automatically manage signing will reset the Code Sign Identity and Provisioning Profile build settings to default values.

Xcode will create and update profiles, app IDs, and certificates

The **Automatically manage signing** setting should now be selected as seen below:



▼ **Signing (Debug)**

☑ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: None

Bundle Identifier: com.cablelabs.micronets.mobile

Provisioning Profile: Xcode Managed Profile

Signing Certificate: Apple Development

Status: ❗ Signing for "Micronets" requires a development team.
Select a development team in the Signing & Capabilities editor.

▼ **Signing (Release)**

☑ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: None

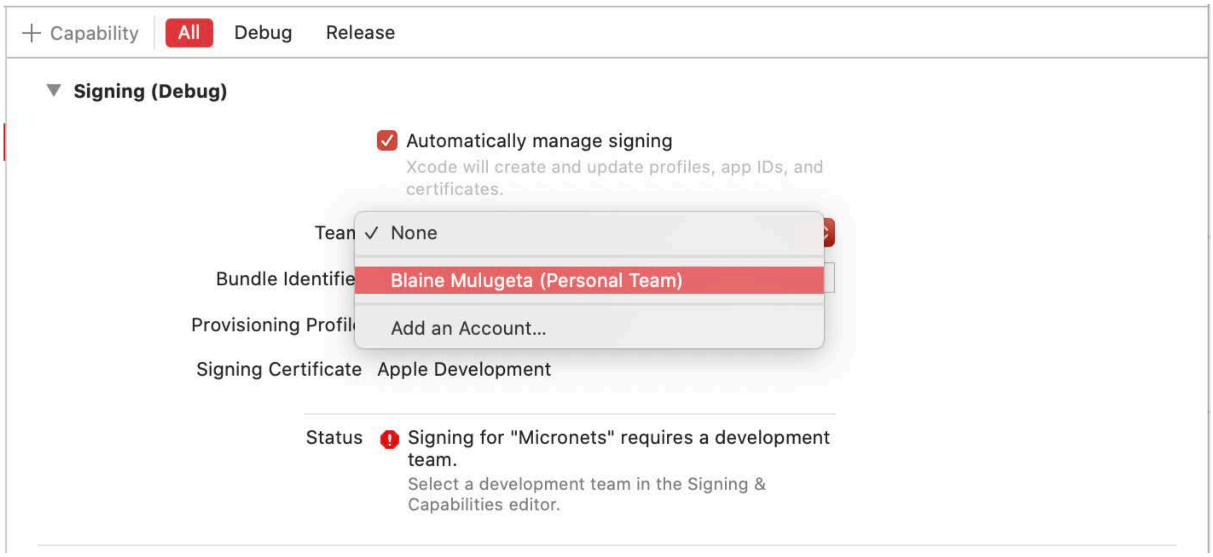
Bundle Identifier: com.cablelabs.micronets.mobile

Provisioning Profile: Xcode Managed Profile

Signing Certificate: Apple Distribution

Status: ❗ Signing for "Micronets" requires a development team.
Select a development team in the Signing & Capabilities editor.

2636 12. Ensure that your team is selected under the **Team** drop-down:



▼ **Signing (Debug)**

☑ Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

Team ✓ None

Blaine Mulugeta (Personal Team)

Add an Account…

Bundle Identifie

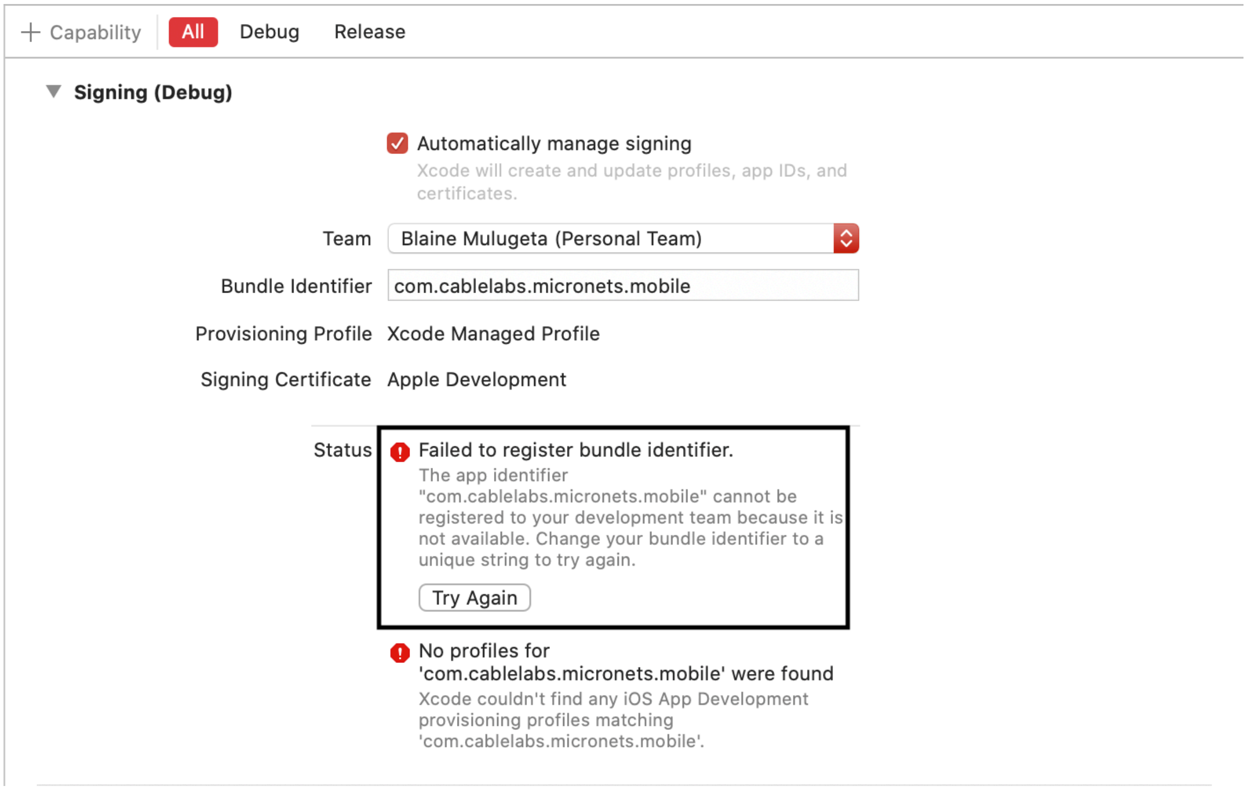
Provisioning Profil

Signing Certificate Apple Development

Status Signing for "Micronets" requires a development team.

Select a development team in the Signing & Capabilities editor.

Note: If you encounter the following error to register the bundle identifier, proceed to step a:



▼ **Signing (Debug)**

☑ Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

Team Blaine Mulugeta (Personal Team)

Bundle Identifier com.cablelabs.micronets.mobile

Provisioning Profile Xcode Managed Profile

Signing Certificate Apple Development

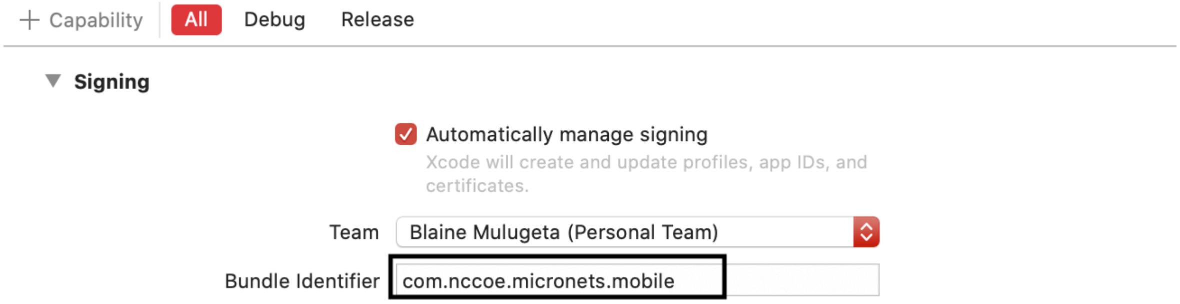
Status Failed to register bundle identifier.

The app identifier "com.cablelabs.micronets.mobile" cannot be registered to your development team because it is not available. Change your bundle identifier to a unique string to try again.

No profiles for 'com.cablelabs.micronets.mobile' were found

Xcode couldn't find any iOS App Development provisioning profiles matching 'com.cablelabs.micronets.mobile'.

a. Change the **Bundle Identifier** to your own unique identifier:



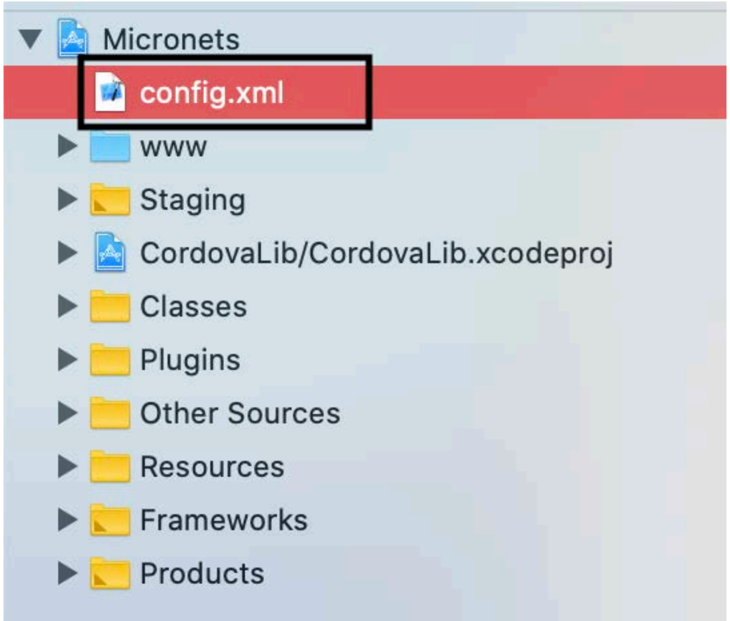
☑ Automatically manage signing

Xcode will create and update profiles, app IDs, and certificates.

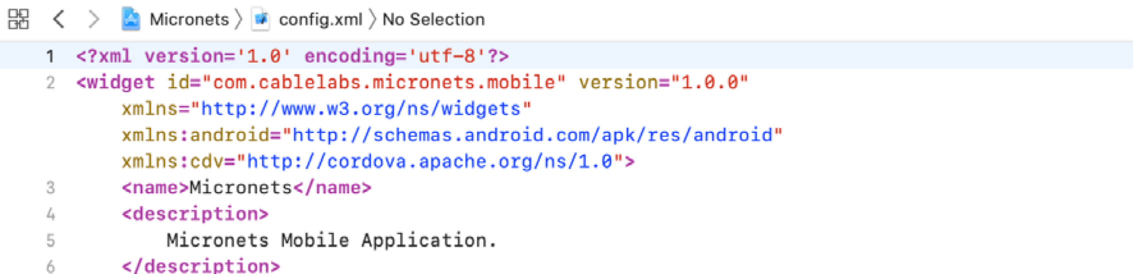
Team: Blaine Mulugeta (Personal Team)

Bundle Identifier: com.nccoe.micronets.mobile

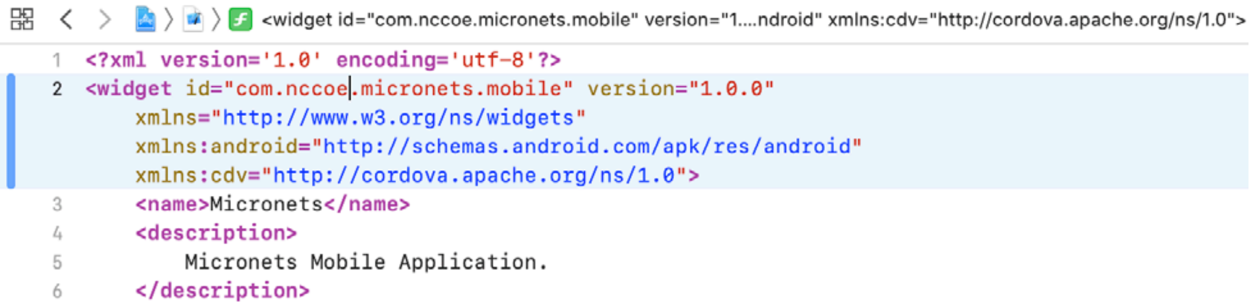
b. Navigate to the ***config.xml*** file by selecting as shown below:



c. Modify the widget id from **com.cablelabs.micronets.mobile** to the build identifier created in step a as seen below:



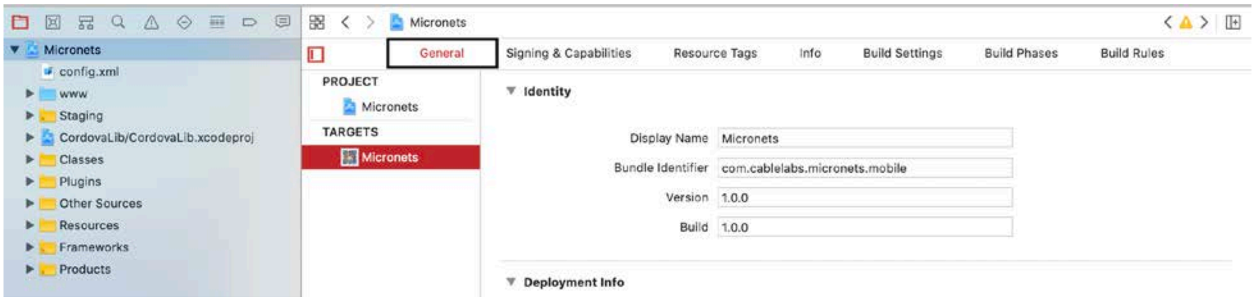
```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.cablelabs.micronets.mobile" version="1.0.0"
    xmlns="http://www.w3.org/ns/widgets"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Micronets</name>
    <description>
        Micronets Mobile Application.
    </description>
```



```
<?xml version='1.0' encoding='utf-8'?>
<widget id="com.nccoe.micronets.mobile" version="1.0.0"
    xmlns="http://www.w3.org/ns/widgets"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Micronets</name>
    <description>
        Micronets Mobile Application.
    </description>
```

2644

2645 13. Select the **General** tab in the heading:

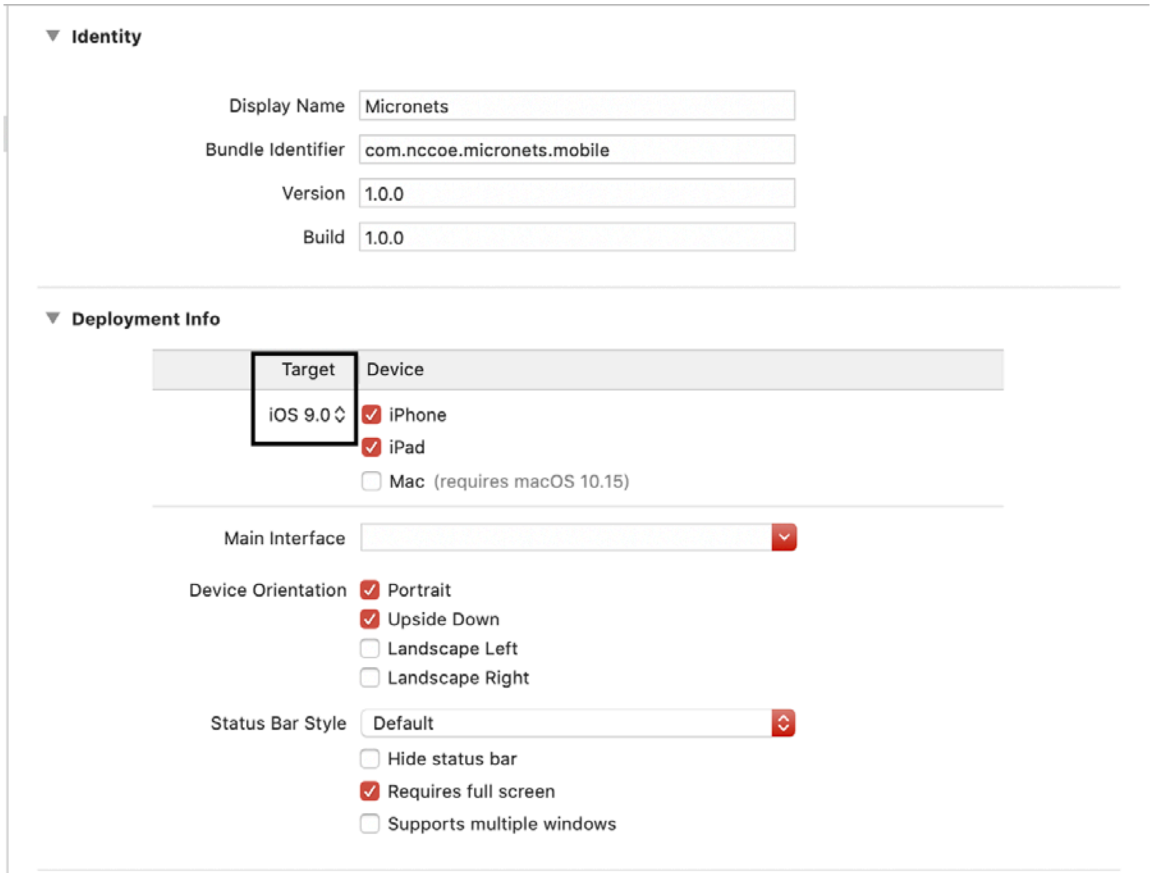


Display Name: Micronets
Bundle Identifier: com.cablelabs.micronets.mobile
Version: 1.0.0
Build: 1.0.0

2646

2647 14. Under **Deployment Info,** make the following modifications:

a. Select the deployment **Target** (suggested 10.3)



▼ Identity

Display Name	Micronets
Bundle Identifier	com.nccoe.micronets.mobile
Version	1.0.0
Build	1.0.0

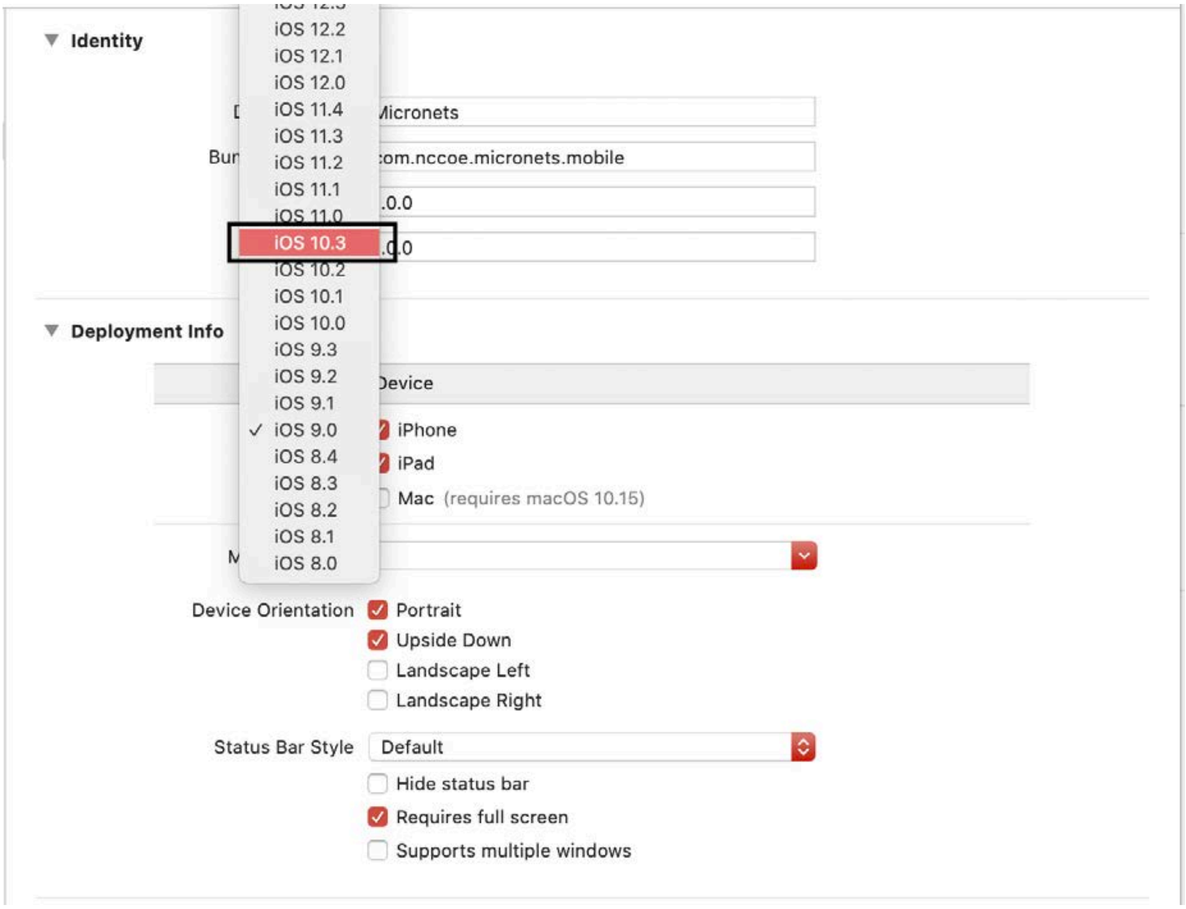
▼ Deployment Info

Target	Device
iOS 9.0	☑ iPhone ☑ iPad □ Mac (requires macOS 10.15)

Main Interface

Device Orientation ☑ Portrait
☑ Upside Down
□ Landscape Left
□ Landscape Right

Status Bar Style Default
□ Hide status bar
☑ Requires full screen
□ Supports multiple windows



▼ Identity

Micronets

Bun om.nccoe.micronets.mobile

.0.0

.0.0

▼ Deployment Info

Device

☑ iPhone

☑ iPad

□ Mac (requires macOS 10.15)

Device Orientation ☑ Portrait

☑ Upside Down

□ Landscape Left

□ Landscape Right

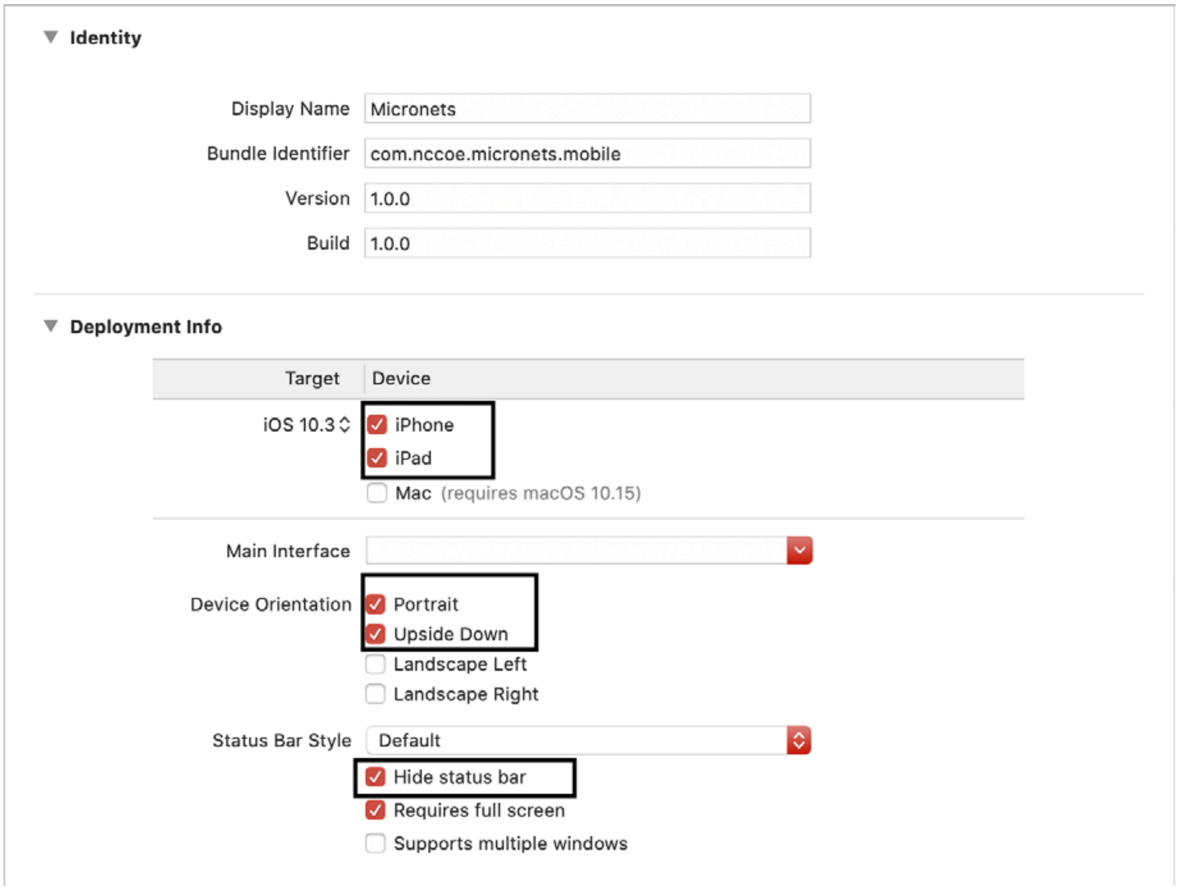
Status Bar Style Default

□ Hide status bar

☑ Requires full screen

□ Supports multiple windows

b. Select Device type **iPhone and iPad,** Device Orientation **Portrait and Upside Down,** Status Bar style **Hide status bar:**



▼ Identity

Display Name: Micronets
Bundle Identifier: com.nccoe.micronets.mobile
Version: 1.0.0
Build: 1.0.0

▼ Deployment Info

Target	Device
iOS 10.3	☑ iPhone ☑ iPad □ Mac (requires macOS 10.15)

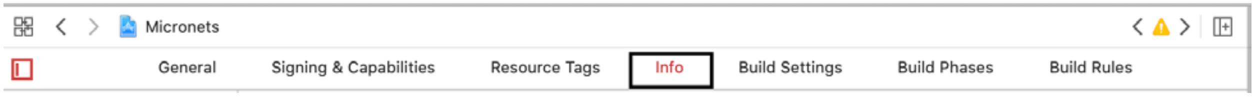
Main Interface:

Device Orientation: ☑ Portrait ☑ Upside Down □ Landscape Left □ Landscape Right

Status Bar Style: Default

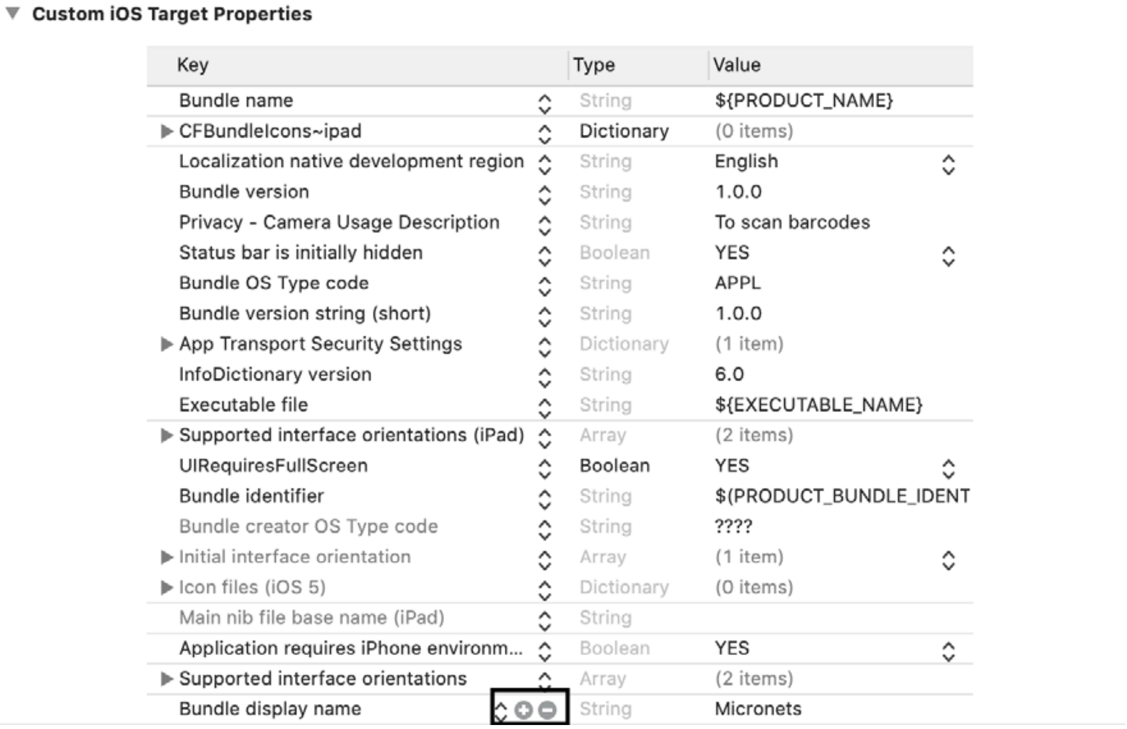
☑ Hide status bar
☑ Requires full screen
□ Supports multiple windows

15. Select the **Info** tab, and make the following modifications:



a. On last entry in **Custom iOS Target Properties,** hover over the down arrow.

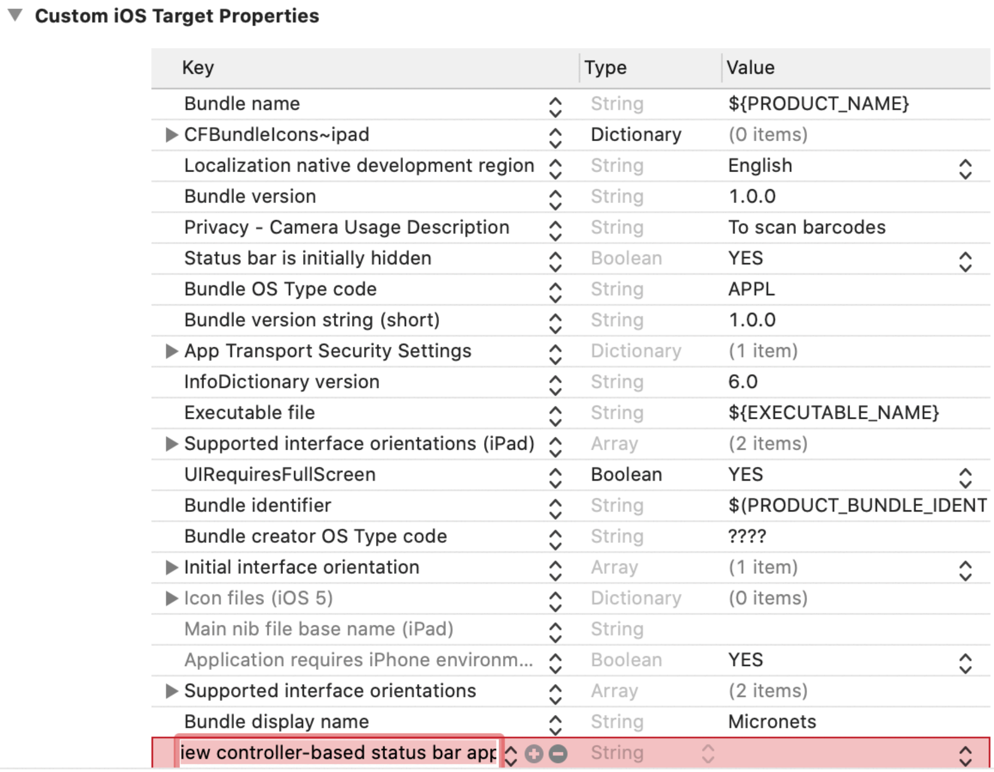
b. A plus sign appears. Click it to create a new property.



Custom iOS Target Properties

Key	Type	Value
Bundle name	String	${PRODUCT_NAME}
▶ CFBundleIcons~ipad	Dictionary	(0 items)
Localization native development region	String	English
Bundle version	String	1.0.0
Privacy - Camera Usage Description	String	To scan barcodes
Status bar is initially hidden	Boolean	YES
Bundle OS Type code	String	APPL
Bundle version string (short)	String	1.0.0
▶ App Transport Security Settings	Dictionary	(1 item)
InfoDictionary version	String	6.0
Executable file	String	${EXECUTABLE_NAME}
▶ Supported interface orientations (iPad)	Array	(2 items)
UIRequiresFullScreen	Boolean	YES
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENT
Bundle creator OS Type code	String	????
▶ Initial interface orientation	Array	(1 item)
▶ Icon files (iOS 5)	Dictionary	(0 items)
Main nib file base name (iPad)	String	
Application requires iPhone environm...	Boolean	YES
▶ Supported interface orientations	Array	(2 items)
Bundle display name	String	Micronets

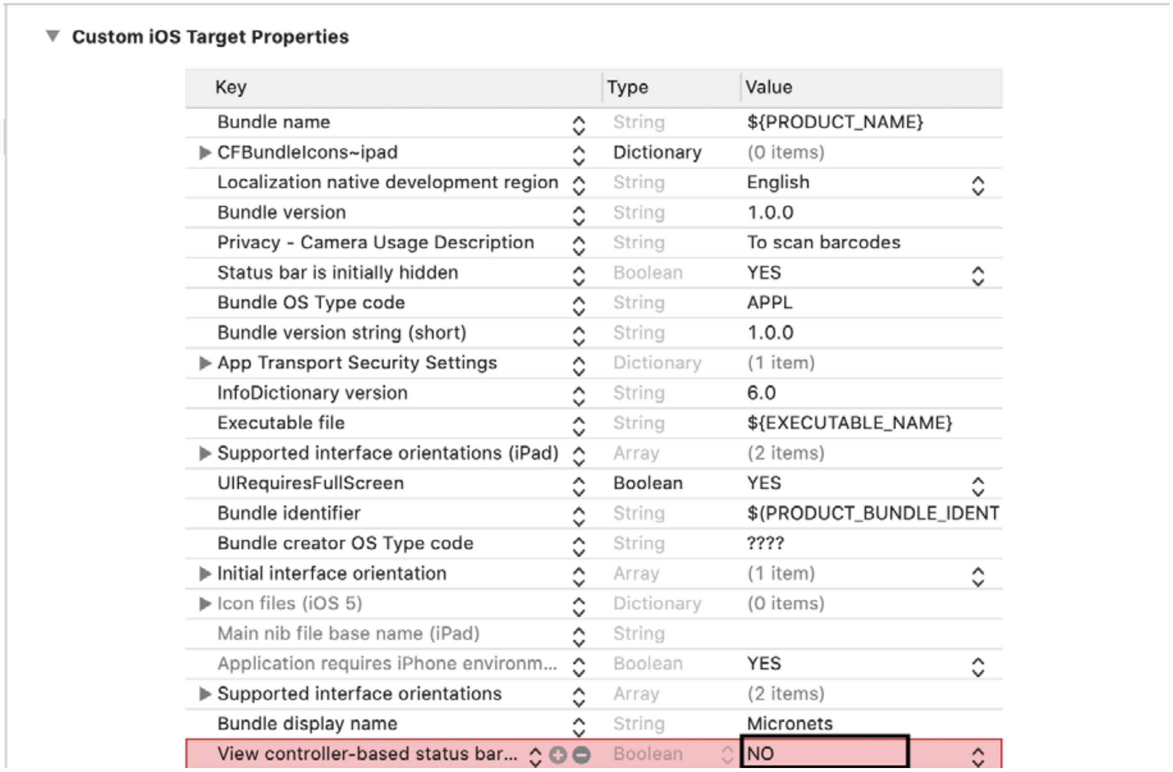
c. In the combo box drop-down, start typing **View controller**, and choose the auto-fill suggestion **View controller-based status bar appearance:**



Custom iOS Target Properties

Key	Type	Value
Bundle name	String	${PRODUCT_NAME}
▶ CFBundleIcons~ipad	Dictionary	(0 items)
Localization native development region	String	English
Bundle version	String	1.0.0
Privacy - Camera Usage Description	String	To scan barcodes
Status bar is initially hidden	Boolean	YES
Bundle OS Type code	String	APPL
Bundle version string (short)	String	1.0.0
▶ App Transport Security Settings	Dictionary	(1 item)
InfoDictionary version	String	6.0
Executable file	String	${EXECUTABLE_NAME}
▶ Supported interface orientations (iPad)	Array	(2 items)
UIRequiresFullScreen	Boolean	YES
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENT
Bundle creator OS Type code	String	????
▶ Initial interface orientation	Array	(1 item)
▶ Icon files (iOS 5)	Dictionary	(0 items)
Main nib file base name (iPad)	String	
Application requires iPhone environm...	Boolean	YES
▶ Supported interface orientations	Array	(2 items)
Bundle display name	String	Micronets
iew controller-based status bar app	String	

d. Click **enter** to add this entry. Ensure this entry is set to **NO.**



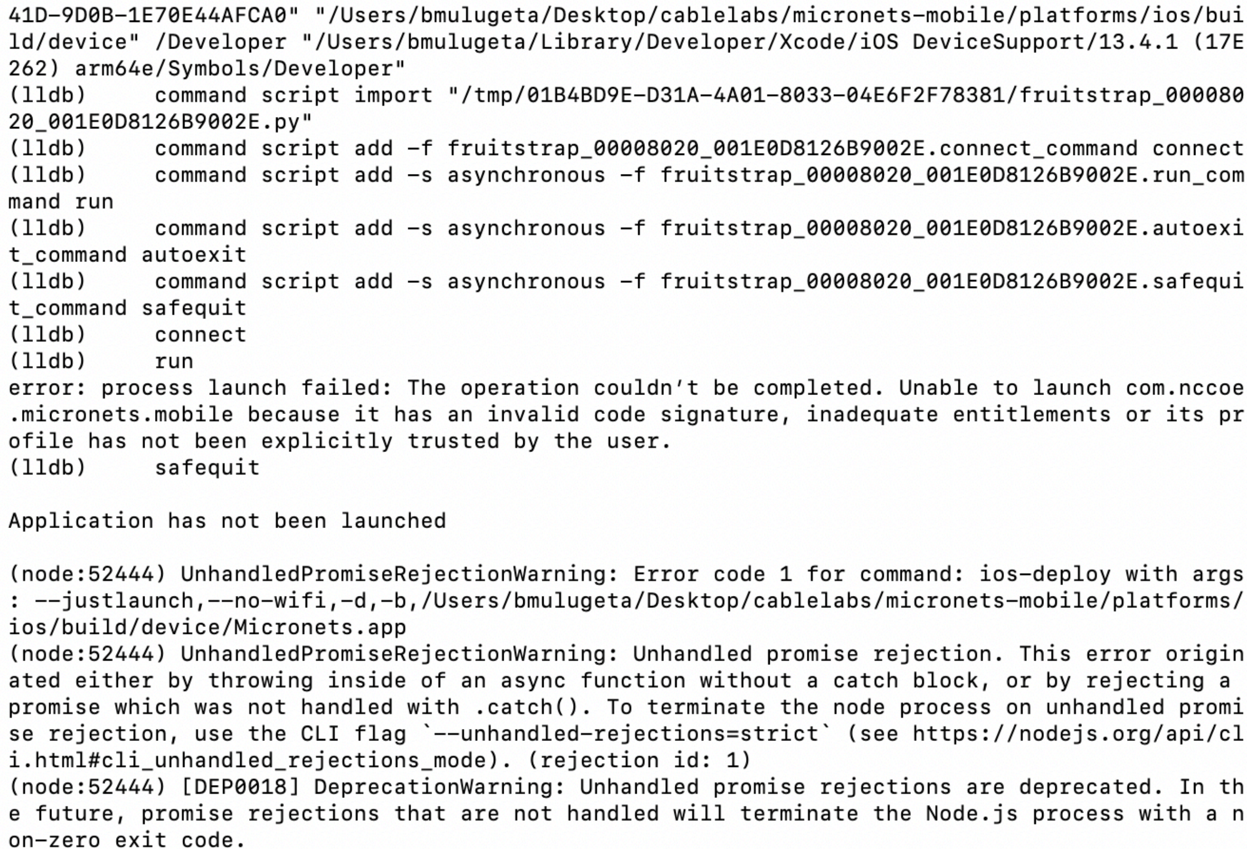
Custom iOS Target Properties

Key	Type	Value
Bundle name	String	${PRODUCT_NAME}
▶ CFBundleIcons~ipad	Dictionary	(0 items)
Localization native development region	String	English
Bundle version	String	1.0.0
Privacy - Camera Usage Description	String	To scan barcodes
Status bar is initially hidden	Boolean	YES
Bundle OS Type code	String	APPL
Bundle version string (short)	String	1.0.0
▶ App Transport Security Settings	Dictionary	(1 item)
InfoDictionary version	String	6.0
Executable file	String	${EXECUTABLE_NAME}
▶ Supported interface orientations (iPad)	Array	(2 items)
UIRequiresFullScreen	Boolean	YES
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENT
Bundle creator OS Type code	String	????
▶ Initial interface orientation	Array	(1 item)
▶ Icon files (iOS 5)	Dictionary	(0 items)
Main nib file base name (iPad)	String	
Application requires iPhone environm...	Boolean	YES
▶ Supported interface orientations	Array	(2 items)
Bundle display name	String	Micronets
View controller-based status bar...	Boolean	NO

16. Return to the terminal, and run the following command (ensure the iPhone is unlocked first):

```
cordova run ios --device --buildFlag='-UseModernBuildSystem=0'
```

Note: You may see an **UnhandledPromiseRejectionWarning** as seen below, but the application should still have been loaded onto your iPhone:



```
41D-9D0B-1E70E44AFCA0" "/Users/bmulugeta/Desktop/cablelabs/micronets-mobile/platforms/ios/bui
ld/device" /Developer "/Users/bmulugeta/Library/Developer/Xcode/iOS DeviceSupport/13.4.1 (17E
262) arm64e/Symbols/Developer"
(lldb)     command script import "/tmp/01B4BD9E-D31A-4A01-8033-04E6F2F78381/fruitstrap_000080
20_001E0D8126B9002E.py"
(lldb)     command script add -f fruitstrap_00008020_001E0D8126B9002E.connect_command connect
(lldb)     command script add -s asynchronous -f fruitstrap_00008020_001E0D8126B9002E.run_com
mand run
(lldb)     command script add -s asynchronous -f fruitstrap_00008020_001E0D8126B9002E.autoexi
t_command autoexit
(lldb)     command script add -s asynchronous -f fruitstrap_00008020_001E0D8126B9002E.safequi
t_command safequit
(lldb)     connect
(lldb)     run
error: process launch failed: The operation couldn’t be completed. Unable to launch com.nccoe
.micronets.mobile because it has an invalid code signature, inadequate entitlements or its pr
ofile has not been explicitly trusted by the user.
(lldb)     safequit

Application has not been launched

(node:52444) UnhandledPromiseRejectionWarning: Error code 1 for command: ios-deploy with args
: --justlaunch,--no-wifi,-d,-b,/Users/bmulugeta/Desktop/cablelabs/micronets-mobile/platforms/
ios/build/device/Micronets.app
(node:52444) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error origin
ated either by throwing inside of an async function without a catch block, or by rejecting a
promise which was not handled with .catch(). To terminate the node process on unhandled promi
se rejection, use the CLI flag `--unhandled-rejections=strict` (see https://nodejs.org/api/cl
i.html#cli_unhandled_rejections_mode). (rejection id: 1)
(node:52444) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In th
e future, promise rejections that are not handled will terminate the Node.js process with a n
on-zero exit code.
```

4.1.12 MSO Portal Bootstrapping Interface to the Onboarding Manager

This section describes the CableLabs Micronets MSO portal, which, for this implementation, is a cloud-provided service. This implementation leverages the nccoe-build-3 branch of CableLabs Micronets MSO portal Git release. This service can be hosted by the implementer or another party. This documentation describes setting up your own MSO portal.

4.1.12.1 MSO Portal Overview

The MSO portal is the interface between the Micronets iPhone application and the Micronets Manager. It is responsible for passing onboarding requests and respective onboarding information to the Micronets Manager to complete the request.

4.1.12.2 Configuration Overview

The following subsections document the software and network configurations for the MSO portal. Please note that the MUD manager, Micronets Manager, Websocket Proxy, MUD registry, and MSO portal are all implemented on the same server, nccoe-server1.micronets.net. Many of these configurations are already covered in previous sections of this document but are repeated here for consistency.

4.1.12.2.1 Network Configuration

This server was hosted outside the lab environment on a Linode cloud-hosted Linux server. Its IP address was statically assigned.

4.1.12.2.2 Software Configuration

The following software is required to install, configure, and operate the MSO portal:

- docker (v18.06 or higher)
- docker-compose (v1.23.1 or higher)
- OpenSSL (1.0.2g or higher)
- NGINX and requisite certificates if https is to be supported

4.1.12.2.3 Hardware Configuration

The following hardware is required to install, configure, and operate the MSO portal:

- 4 GB of RAM
- 50 GB of free disk space

4.1.12.3 Setup

4.1.12.3.1 Install Dependencies

1. Install docker, docker-compose, openssl, and NGINX by entering the following command:

```
sudo apt-get install docker docker-compose openssl nginx
```

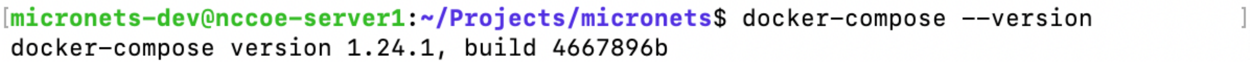
4.1.12.3.2 Install and Configure MSO Portal

1. Install a newer version of docker-compose, if necessary. (Ubuntu 18.04 comes with an older version.)

    a. Check the current version by entering the following command:

    ```
    docker-compose --version
    ```

    The result should be similar to the following:



```
micronets-dev@nccoe-server1:~/Projects/micronets$ docker-compose --version
docker-compose version 1.24.1, build 4667896b
```

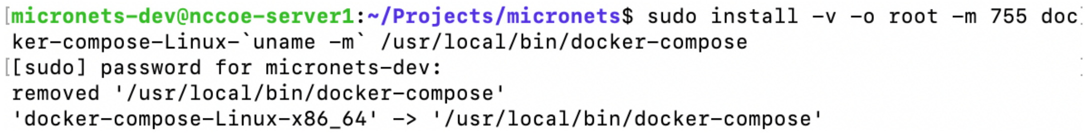
    b. If the version is earlier than v1.23.1, run the following commands to install a new version in /usr/local/bin:
        i. Download the docker compose utility:

           ```
           curl -L -O
           https://github.com/docker/compose/releases/download/1.24.1/docker-
           compose-Linux-`uname -m’
           ```

        ii. Install the docker compose utility into the appropriate directory:

            ```
            sudo install -v -o root -m 755 docker-compose-Linux-`uname -m`
            /usr/local/bin/docker-compose
            ```

            The result should be similar to the following:



```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 doc
 ker-compose-Linux-`uname -m` /usr/local/bin/docker-compose
[[sudo] password for micronets-dev:
 removed '/usr/local/bin/docker-compose'
 'docker-compose-Linux-x86_64' -> '/usr/local/bin/docker-compose'
```

2. Download and install the MSO portal management script by entering the following commands:
    a. Download the MSO portal management script by executing the following command:

       ```
       curl -O https://raw.githubusercontent.com/cablelabs/micronets-mso-
       portal/nccoe-build-3/scripts/mso-portal
       ```

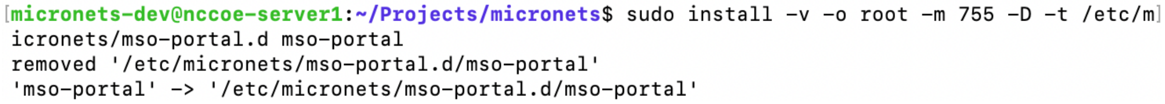
    b. Download the *docker-compose.yml* file by executing the following command:

       ```
       curl -O https://raw.githubusercontent.com/cablelabs/micronets-mso-
       portal/nccoe-build-3/scripts/docker-compose.yml
       ```

    c. Install the MSO portal management script to the appropriate directory by executing the following command:

       ```
       sudo install -v -o root -m 755 -D -t /etc/micronets/mso-portal.d mso-
       portal
       ```

       The result should be similar to the following:

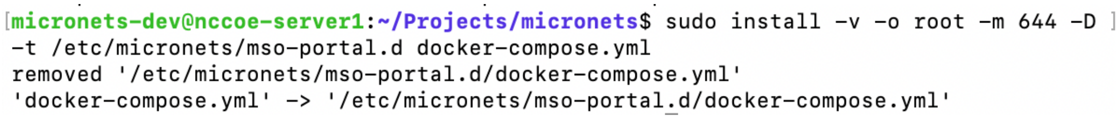


```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 755 -D -t /etc/m]
icronets/mso-portal.d mso-portal
removed '/etc/micronets/mso-portal.d/mso-portal'
'mso-portal' -> '/etc/micronets/mso-portal.d/mso-portal'
```

d. Install the *docker-compose.yml* management script to the appropriate directory by executing the following command:

```
sudo install -v -o root -m 644 -D -t /etc/micronets/mso-portal.d docker-
compose.yml
```

The result should be similar to the following:



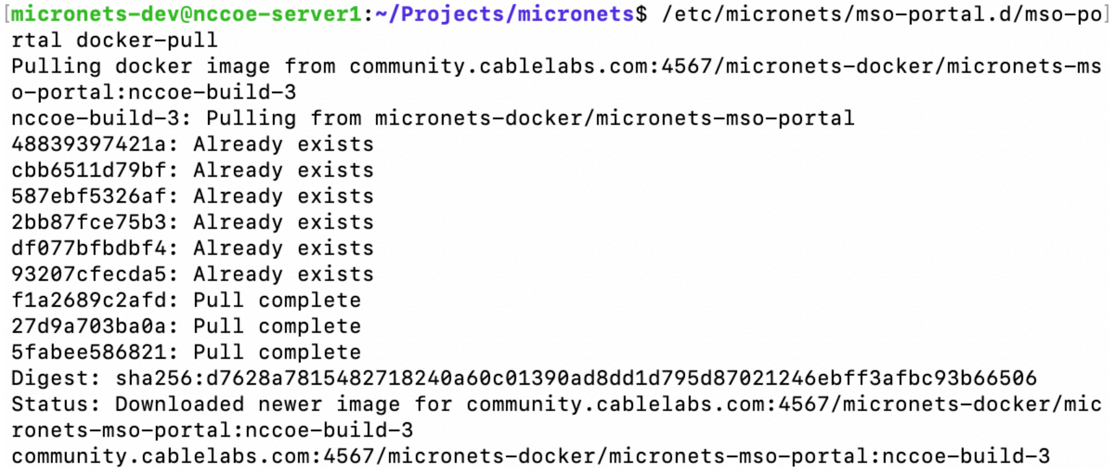
```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo install -v -o root -m 644 -D ]
-t /etc/micronets/mso-portal.d docker-compose.yml
removed '/etc/micronets/mso-portal.d/docker-compose.yml'
'docker-compose.yml' -> '/etc/micronets/mso-portal.d/docker-compose.yml'
```

Note: The MSO portal management script contains default values that can be modified directly in your copy of the management script or overridden via command-line parameters. Run /etc/micronets/mso-portal.d --help to see the options.

3. Download the MSO portal docker image by executing the following command (Note: If you cannot connect to the docker service, you can use sudo usermod -aG docker <username> to add the user account to the docker group):

```
/etc/micronets/mso-portal.d/mso-portal docker-pull
```

The result should be similar to the following:

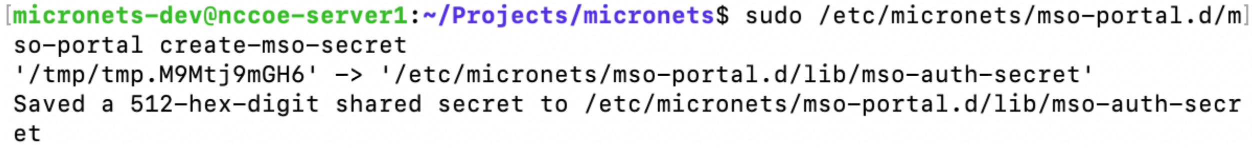


```
micronets-dev@nccoe-server1:~/Projects/micronets$ /etc/micronets/mso-portal.d/mso-po
rtal docker-pull
Pulling docker image from community.cablelabs.com:4567/micronets-docker/micronets-ms
o-portal:nccoe-build-3
nccoe-build-3: Pulling from micronets-docker/micronets-mso-portal
48839397421a: Already exists
cbb6511d79bf: Already exists
587ebf5326af: Already exists
2bb87fce75b3: Already exists
df077bfbdbf4: Already exists
93207cfecda5: Already exists
f1a2689c2afd: Pull complete
27d9a703ba0a: Pull complete
5fabee586821: Pull complete
Digest: sha256:d7628a7815482718240a60c01390ad8dd1d795d87021246ebff3afbc93b66506
Status: Downloaded newer image for community.cablelabs.com:4567/micronets-docker/mic
ronets-mso-portal:nccoe-build-3
community.cablelabs.com:4567/micronets-docker/micronets-mso-portal:nccoe-build-3
```

4. Generate a shared secret for enabling communication between the Micronets Manager instances and the MSO portal:

   ```
   sudo /etc/micronets/mso-portal.d/mso-portal create-mso-secret
   ```

   The result should be similar to the following:

   ```
   micronets-dev@nccoe-server1:~/Projects/micronets$ sudo /etc/micronets/mso-portal.d/m
   so-portal create-mso-secret
   '/tmp/tmp.M9Mtj9mGH6' -> '/etc/micronets/mso-portal.d/lib/mso-auth-secret'
   Saved a 512-hex-digit shared secret to /etc/micronets/mso-portal.d/lib/mso-auth-secr
   et
   ```

   Note: This value will need to be copied to the Micronets Manager host server to allow Micronets Manager instances to access the MSO portal APIs.

5. Configure MSO portal URLs:

   a. Open the *mso-portal* file by entering the following command:

      ```
      sudo vim /etc/micronets/mso-portal.d/mso-portal
      ```

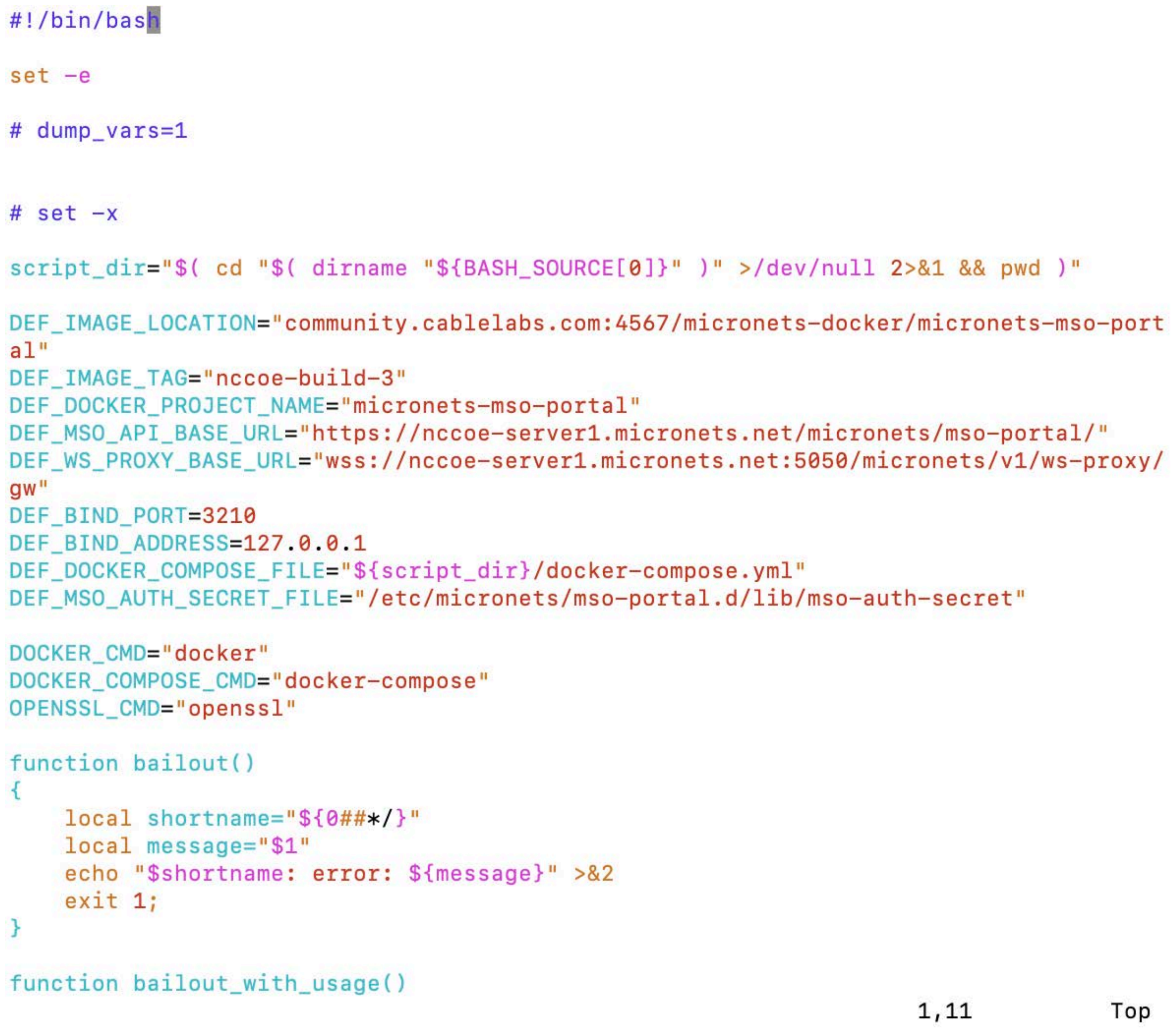
   b. Modify the parameters of the MSO portal management script to reflect the public end points of the MSO portal service. For example:

      i. The **DEF_MSO_API_BASE_URL** path variable can be set to:

         ```
         DEF_MSO_API_BASE_URL="https://nccoe-
         server1.micronets.net/micronets/mso-portal/"
         ```

      ii. The **DEF_WS_PROXY_BASE_URL** path variable can be set to:

```
DEF_WS_PROXY_BASE_URL="wss:// nccoe-server1.micronets.net:5050/micronets/v1/ws-proxy/gw"
```

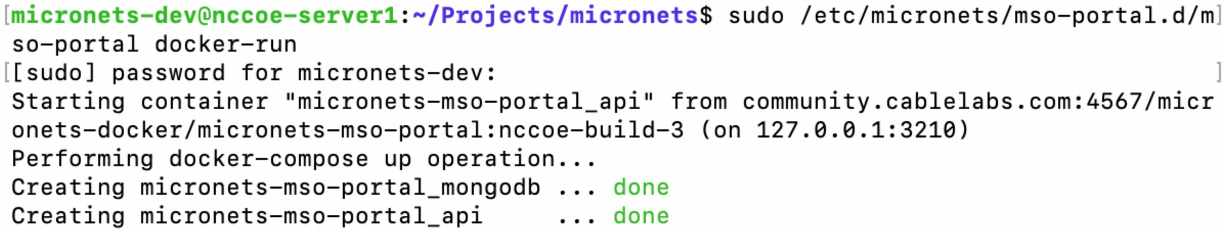


```
#!/bin/bash

set -e

# dump_vars=1


# set -x

script_dir="$( cd "$( dirname "${BASH_SOURCE[0]}" )" >/dev/null 2>&1 && pwd )"

DEF_IMAGE_LOCATION="community.cablelabs.com:4567/micronets-docker/micronets-mso-port
al"
DEF_IMAGE_TAG="nccoe-build-3"
DEF_DOCKER_PROJECT_NAME="micronets-mso-portal"
DEF_MSO_API_BASE_URL="https://nccoe-server1.micronets.net/micronets/mso-portal/"
DEF_WS_PROXY_BASE_URL="wss://nccoe-server1.micronets.net:5050/micronets/v1/ws-proxy/
gw"
DEF_BIND_PORT=3210
DEF_BIND_ADDRESS=127.0.0.1
DEF_DOCKER_COMPOSE_FILE="${script_dir}/docker-compose.yml"
DEF_MSO_AUTH_SECRET_FILE="/etc/micronets/mso-portal.d/lib/mso-auth-secret"

DOCKER_CMD="docker"
DOCKER_COMPOSE_CMD="docker-compose"
OPENSSL_CMD="openssl"

function bailout()
{
    local shortname="${0##*/}"
    local message="$1"
    echo "$shortname: error: ${message}" >&2
    exit 1;
}

function bailout_with_usage()
                                                                      1,11          Top
```

6. Start the MSO portal docker image by executing the following command:

```
sudo /etc/micronets/mso-portal.d/mso-portal docker-run
```

The result should be similar to the following:



```
[micronets-dev@nccoe-server1:~/Projects/micronets$ sudo /etc/micronets/mso-portal.d/m]
 so-portal docker-run
[[sudo] password for micronets-dev:                                                  ]
 Starting container "micronets-mso-portal_api" from community.cablelabs.com:4567/micr
 onets-docker/micronets-mso-portal:nccoe-build-3 (on 127.0.0.1:3210)
 Performing docker-compose up operation...
 Creating micronets-mso-portal_mongodb ... done
 Creating micronets-mso-portal_api     ... done
```

7. Verify that the MSO portal started successfully by executing the following command:

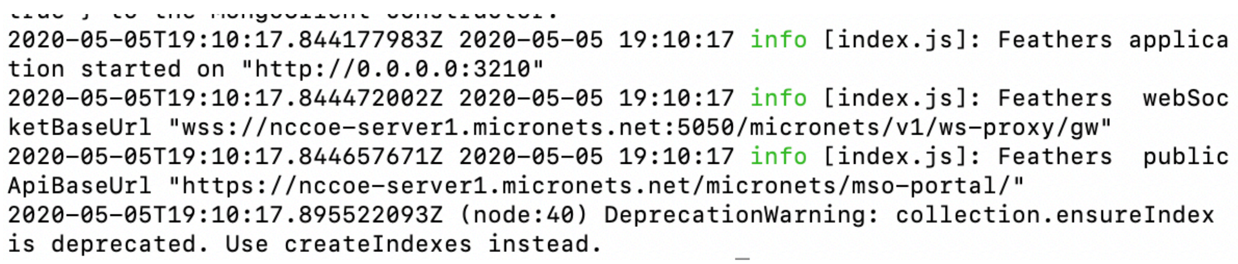
   ```
   /etc/micronets/mso-portal.d/mso-portal docker-logs
   ```

   You should see output like the following at the end of the log:

   Feathers application started on "http://0.0.0.0:3210"

   Feathers   webSocketBaseUrl "wss://<ServerURL>:5050/micronets/v1/ws-proxy/gw"

   Feathers   publicApiBaseUrl "https://< ServerURL>/micronets/mso-portal/"



```
true } to the MongoClient constructor.
2020-05-05T19:10:17.844177983Z 2020-05-05 19:10:17 info [index.js]: Feathers applica
tion started on "http://0.0.0.0:3210"
2020-05-05T19:10:17.844472002Z 2020-05-05 19:10:17 info [index.js]: Feathers  webSoc
ketBaseUrl "wss://nccoe-server1.micronets.net:5050/micronets/v1/ws-proxy/gw"
2020-05-05T19:10:17.844657671Z 2020-05-05 19:10:17 info [index.js]: Feathers  public
ApiBaseUrl "https://nccoe-server1.micronets.net/micronets/mso-portal/"
2020-05-05T19:10:17.895522093Z (node:40) DeprecationWarning: collection.ensureIndex
is deprecated. Use createIndexes instead.
```

8. To securely expose the MSO API, configure your NGINX server block to allow the https proxy to redirect to localhost port 3210:

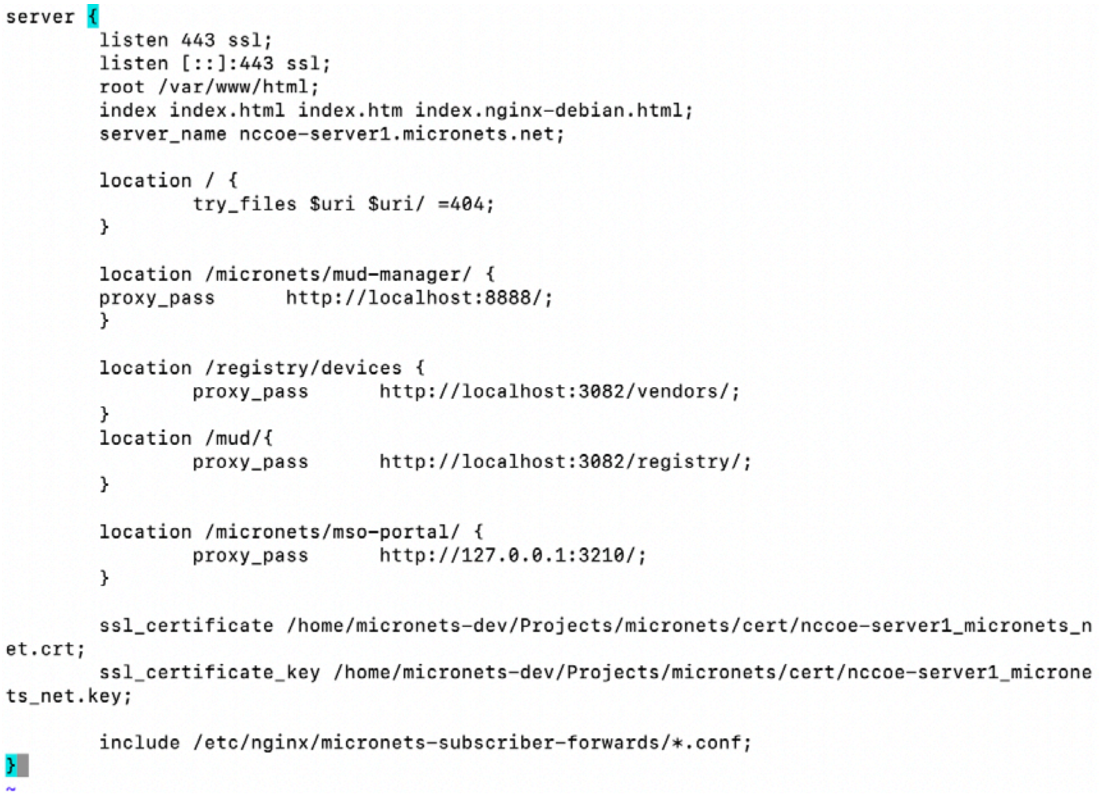
   a. Open the NGINX sites-available file for the server:

      ```
      sudo vim /etc/nginx/sites-available/nccoe-server1.micronets.net
      ```

   b. Add the following location to the server block:

      ```
      server {
             […]
          location /micronets/mso-portal/ {
             proxy_pass   http://127.0.0.1:3210/;
          }
      ```

```
                [...]
        }
```



```
server {
        listen 443 ssl;
        listen [::]:443 ssl;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;
        server_name nccoe-server1.micronets.net;

        location / {
                try_files $uri $uri/ =404;
        }

        location /micronets/mud-manager/ {
        proxy_pass      http://localhost:8888/;
        }

        location /registry/devices {
                proxy_pass      http://localhost:3082/vendors/;
        }
        location /mud/{
                proxy_pass      http://localhost:3082/registry/;
        }

        location /micronets/mso-portal/ {
                proxy_pass      http://127.0.0.1:3210/;
        }

        ssl_certificate /home/micronets-dev/Projects/micronets/cert/nccoe-server1_micronets_n
et.crt;
        ssl_certificate_key /home/micronets-dev/Projects/micronets/cert/nccoe-server1_microne
ts_net.key;

        include /etc/nginx/micronets-subscriber-forwards/*.conf;
}
~
```

4.2 Product Integration and Operation

This section details integration and operation of the Micronets components that were previously installed in the product installation section. Please ensure that the components from that section are installed as described before proceeding to the following sections.

4.2.1 Adding an MSO Subscriber

This section describes adding an MSO portal subscriber. This subscriber account will allow a valid connection and association among the Micronets mobile application, Micronets Gateway, and Micronets services.

4.2.1.1 Prerequisites

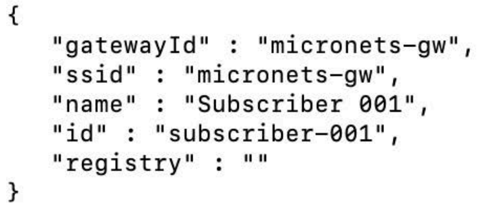
To successfully complete this section, complete the product installation section.

4.2.1.2 Instructions

1. Add a subscriber and associated user account and password to the MSO portal by entering the following command (Note: Be sure to use the server URL that reflects the location of your MSO portal):

```
curl -s -X POST https://nccoe-server1.micronets.net/micronets/mso-
portal/portal/v1/subscriber \
   -H "Content-Type: application/json" \
   -d '{
          "id" : "subscriber-001",
          "ssid" : "micronets-gw",
          "name" : "Subscriber 001",
          "gatewayId":"micronets-gw",
          "username":"micronets",
          "password":"micronets"
   }' \
| json_pp
```

You should see output similar to the following:

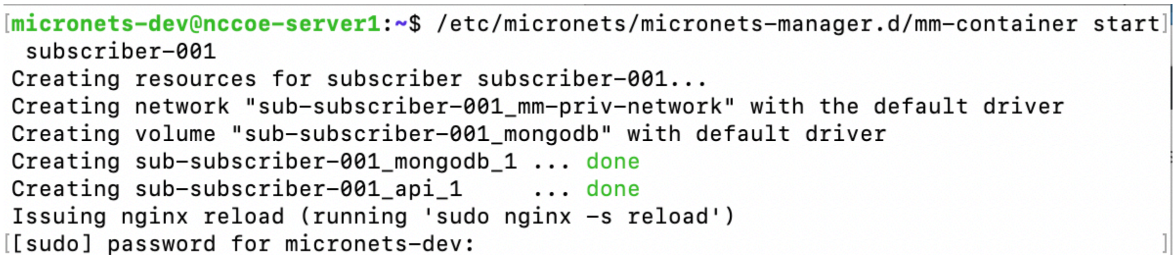


```
{
   "gatewayId" : "micronets-gw",
   "ssid" : "micronets-gw",
   "name" : "Subscriber 001",
   "id" : "subscriber-001",
   "registry" : ""
}
```

2. Start the Micronets Manager for the subscriber by executing the following command:

```
sudo /etc/micronets/micronets-manager.d/mm-container start subscriber-001
```

You should see output similar to the following:

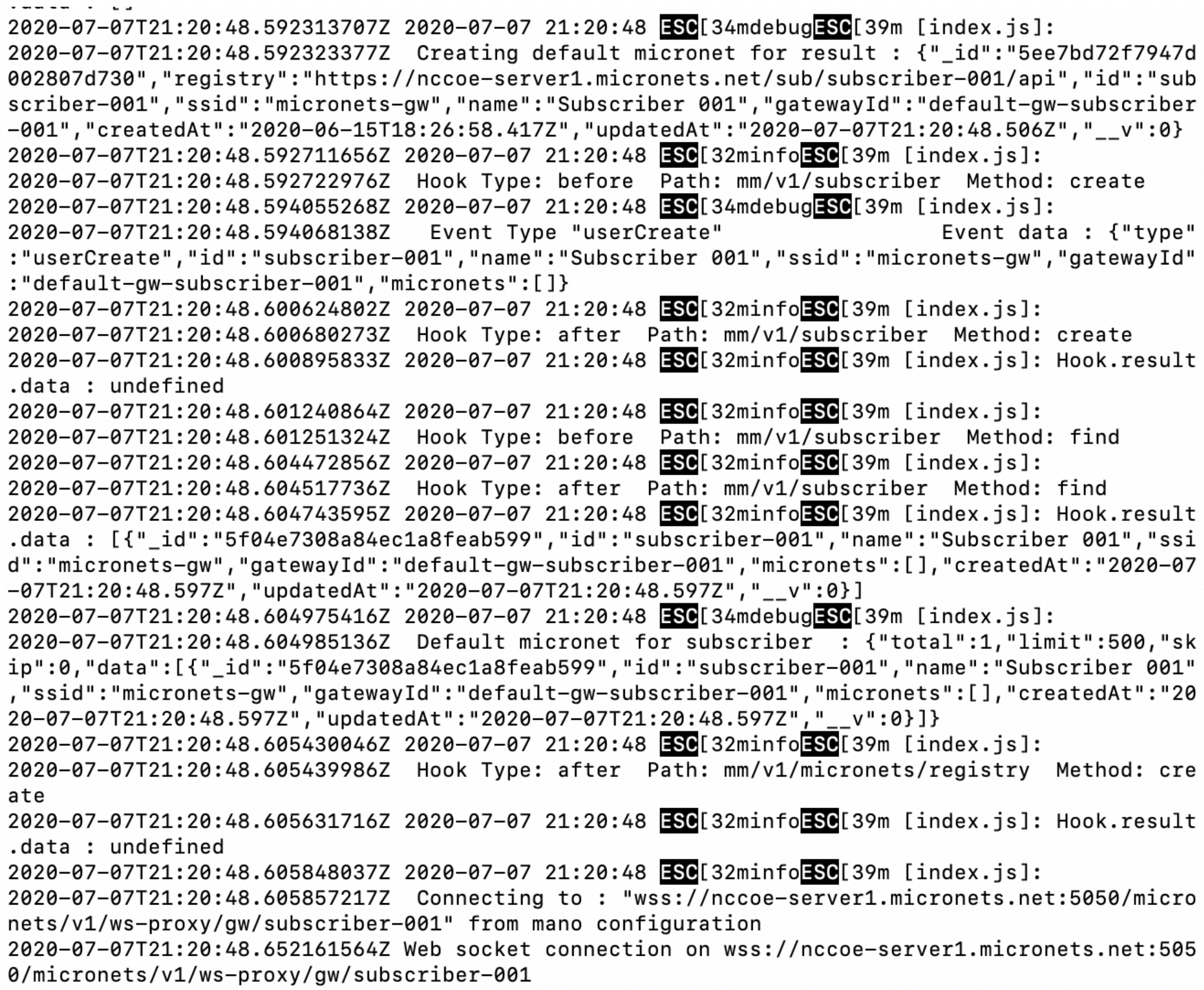


```
micronets-dev@nccoe-server1:~$ /etc/micronets/micronets-manager.d/mm-container start
 subscriber-001
Creating resources for subscriber subscriber-001...
Creating network "sub-subscriber-001_mm-priv-network" with the default driver
Creating volume "sub-subscriber-001_mongodb" with default driver
Creating sub-subscriber-001_mongodb_1 ... done
Creating sub-subscriber-001_api_1     ... done
Issuing nginx reload (running 'sudo nginx -s reload')
[sudo] password for micronets-dev:
```

3. Check the logs to confirm that the Micronets Manager for the new subscriber started successfully by executing the following command:

```
/etc/micronets/micronets-manager.d/mm-container logs subscriber-001
```

You should see output similar to the following:

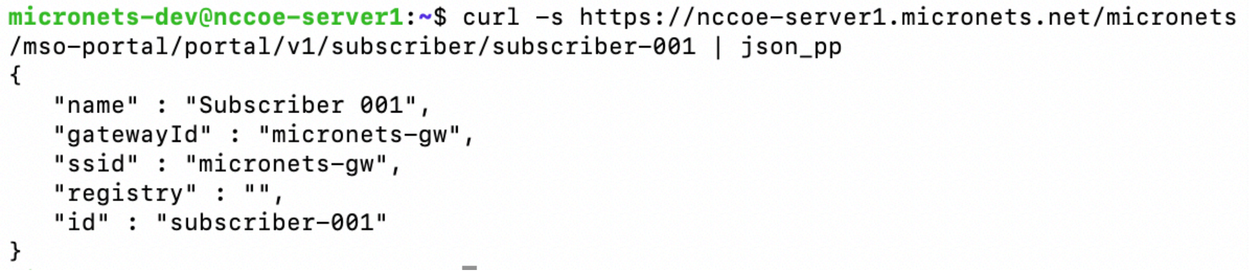


```
2020-07-07T21:20:48.592313707Z 2020-07-07 21:20:48 ESC[34mdebugESC[39m [index.js]:
2020-07-07T21:20:48.592323377Z  Creating default micronet for result : {"_id":"5ee7bd72f7947d
002807d730","registry":"https://nccoe-server1.micronets.net/sub/subscriber-001/api","id":"sub
scriber-001","ssid":"micronets-gw","name":"Subscriber 001","gatewayId":"default-gw-subscriber
-001","createdAt":"2020-06-15T18:26:58.417Z","updatedAt":"2020-07-07T21:20:48.506Z","__v":0}
2020-07-07T21:20:48.592711656Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.592722976Z  Hook Type: before  Path: mm/v1/subscriber  Method: create
2020-07-07T21:20:48.594055268Z 2020-07-07 21:20:48 ESC[34mdebugESC[39m [index.js]:
2020-07-07T21:20:48.594068138Z   Event Type "userCreate"                 Event data : {"type"
:"userCreate","id":"subscriber-001","name":"Subscriber 001","ssid":"micronets-gw","gatewayId"
:"default-gw-subscriber-001","micronets":[]}
2020-07-07T21:20:48.600624802Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.600680273Z  Hook Type: after  Path: mm/v1/subscriber  Method: create
2020-07-07T21:20:48.600895833Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]: Hook.result
.data : undefined
2020-07-07T21:20:48.601240864Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.601251324Z  Hook Type: before  Path: mm/v1/subscriber  Method: find
2020-07-07T21:20:48.604472856Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.604517736Z  Hook Type: after  Path: mm/v1/subscriber  Method: find
2020-07-07T21:20:48.604743595Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]: Hook.result
.data : [{"_id":"5f04e7308a84ec1a8feab599","id":"subscriber-001","name":"Subscriber 001","ssi
d":"micronets-gw","gatewayId":"default-gw-subscriber-001","micronets":[],"createdAt":"2020-07
-07T21:20:48.597Z","updatedAt":"2020-07-07T21:20:48.597Z","__v":0}]
2020-07-07T21:20:48.604975416Z 2020-07-07 21:20:48 ESC[34mdebugESC[39m [index.js]:
2020-07-07T21:20:48.604985136Z  Default micronet for subscriber  : {"total":1,"limit":500,"sk
ip":0,"data":[{"_id":"5f04e7308a84ec1a8feab599","id":"subscriber-001","name":"Subscriber 001"
,"ssid":"micronets-gw","gatewayId":"default-gw-subscriber-001","micronets":[],"createdAt":"20
20-07-07T21:20:48.597Z","updatedAt":"2020-07-07T21:20:48.597Z","__v":0}]}
2020-07-07T21:20:48.605430046Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.605439986Z  Hook Type: after  Path: mm/v1/micronets/registry  Method: cre
ate
2020-07-07T21:20:48.605631716Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]: Hook.result
.data : undefined
2020-07-07T21:20:48.605848037Z 2020-07-07 21:20:48 ESC[32minfoESC[39m [index.js]:
2020-07-07T21:20:48.605857217Z  Connecting to : "wss://nccoe-server1.micronets.net:5050/micro
nets/v1/ws-proxy/gw/subscriber-001" from mano configuration
2020-07-07T21:20:48.652161564Z Web socket connection on wss://nccoe-server1.micronets.net:505
0/micronets/v1/ws-proxy/gw/subscriber-001
```

4. Verify that the Micronets Manager for the subscriber has registered with the MSO portal by executing the following command:

```
curl -s https://my-server.org/micronets/mso-
portal/portal/v1/subscriber/subscriber-001 | json_pp
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -s https://nccoe-server1.micronets.net/micronets
/mso-portal/portal/v1/subscriber/subscriber-001 | json_pp
{
   "name" : "Subscriber 001",
   "gatewayId" : "micronets-gw",
   "ssid" : "micronets-gw",
   "registry" : "",
   "id" : "subscriber-001"
}
```

4.2.2 Associating the Micronets Gateway with a Subscriber

This section describes associating an MSO portal subscriber with the Micronets Gateway. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation: https://github.com/cablelabs/micronets-gw/releases/tag/1.0.62-u18.04 (for Micronets Gateway configuration) and https://github.com/cablelabs/micronets/blob/nccoe-build-3/docs/operation/gateway-4subscriber.md (for operations documentation).

4.2.2.1 Prerequisites

To successfully complete this section, complete the product installation section and complete Section 4.2.1. Ensure that all steps have been successfully completed before proceeding to the instructions.

4.2.2.2 Instructions

1. Create the /etc/network/interfaces file on the Micronets Gateway:

   a. Open a terminal on the Micronets Gateway. If this is the first installation of the Micronets Gateway, copy the sample interfaces file to your /etc/network/interfaces file by entering the following command:
   ```
   sudo cp /opt/micronets-gw/doc/interfaces.sample /etc/network/interfaces
   ```

   Modify the /etc/network/interfaces file:

   Retrieve the desired interface names on the gateway by running the following command in a terminal on the gateway:

   ```
   ifconfig
   ```

   Configure your wireless and wired interface by renaming the corresponding portion of the file to reference the respective interface name as seen in the config below:

   ```
   #
   # A wired interface managed by the Micronets gateway
   ```

```
#
allow-brmn001 enp1s0
iface enp1s0 manual
  ovs_type OVSPort
  ovs_bridge brmn001
  ovs_port_req 4
  ovs_port_initial_state blocked
#
# A wireless interface managed by the Micronets gateway
#
allow-brmn001 wlp2s0
iface wlp2s0inet manual
  ovs_type OVSPort
  ovs_bridge brmn001
  ovs_port_req 3
  ovs_port_initial_state blocked
```

Confirm that the bridge entry contains an ovs_ports line referring to the micronet interfaces **(enp1s0** and **wlp2s0)** as seen in the config below:

```
auto brmn001
allow-ovs brmn001
iface brmn001 inet manual
 ovs_type OVSBridge
  ...
 # the ovs_ports should list all wired and wireless interfaces under Micronets management
  ovs_ports diagout1 enp1s0 wlp2s0
  ...
```

Confirm that the entry in the interfaces file for the wired interface is set up correctly for the network to supply the uplink (the uplink interface is enp1s0) and get its address via DHCP so the configuration is similar to the following:

```
#
# The uplink port
```

```
#
auto eth enp1s0
iface eth0inet dhcp
```

Confirm that the bridge entry contains an **ovs_bridge_uplink_port** line referring to the uplink interface as seen in the config below:

```
auto brmn001
allow-ovs brmn001
iface brmn001 inet manual
  ovs_type OVSBridge
  ...
  # This is the port that's connected to the Internet
  ovs_bridge_uplink_port enp1s0
  ...
```

Reboot the gateway to apply the changes to the /etc/network/interfaces file by executing the following command:

```
sudo reboot
```

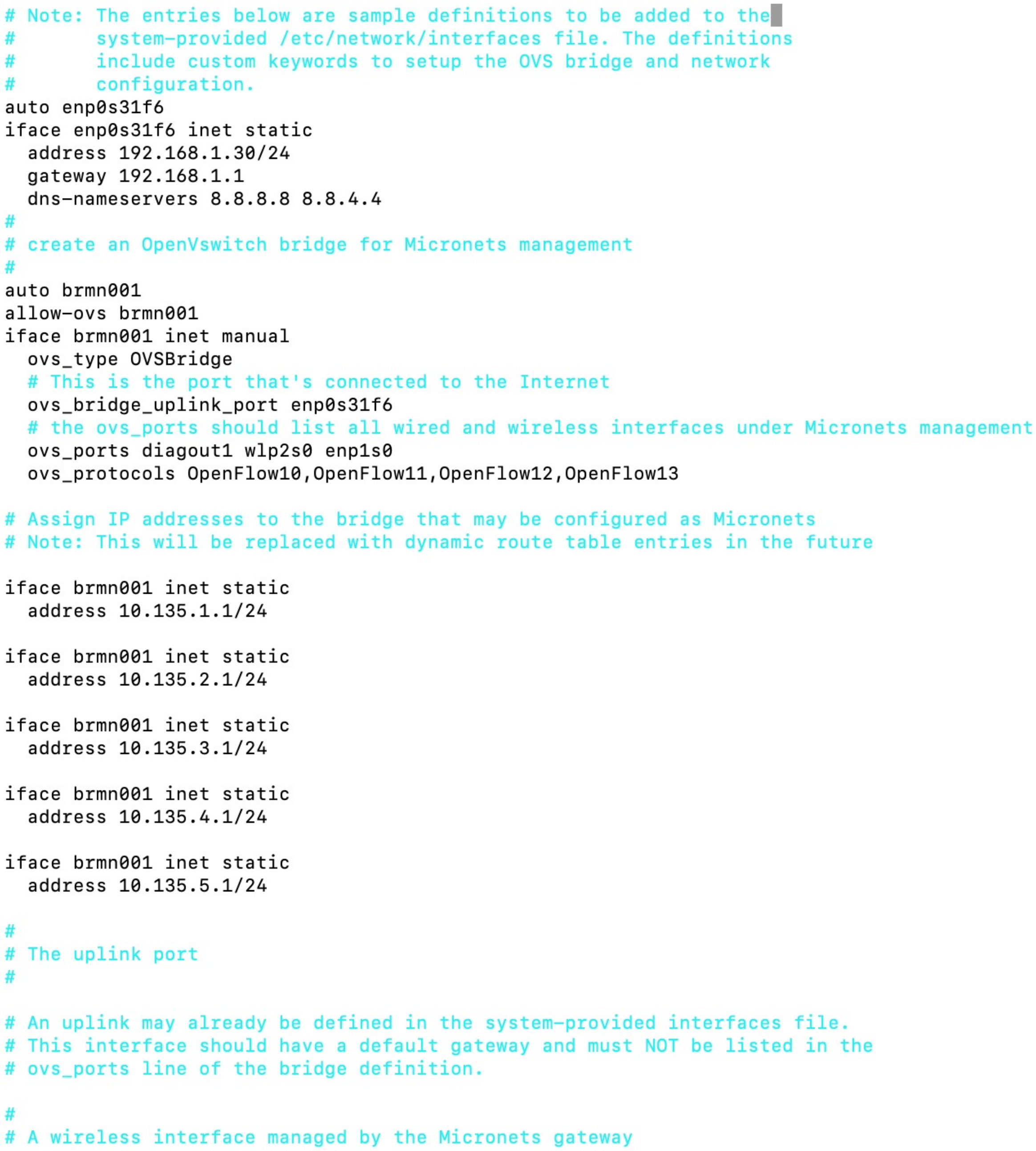
2. Create a gateway configuration file for the Micronets Gateway to register for the subscriber:
   a. Copy and save the MAC addresses and corresponding interface names output by executing the following command:

      ```
      ifconfig
      ```

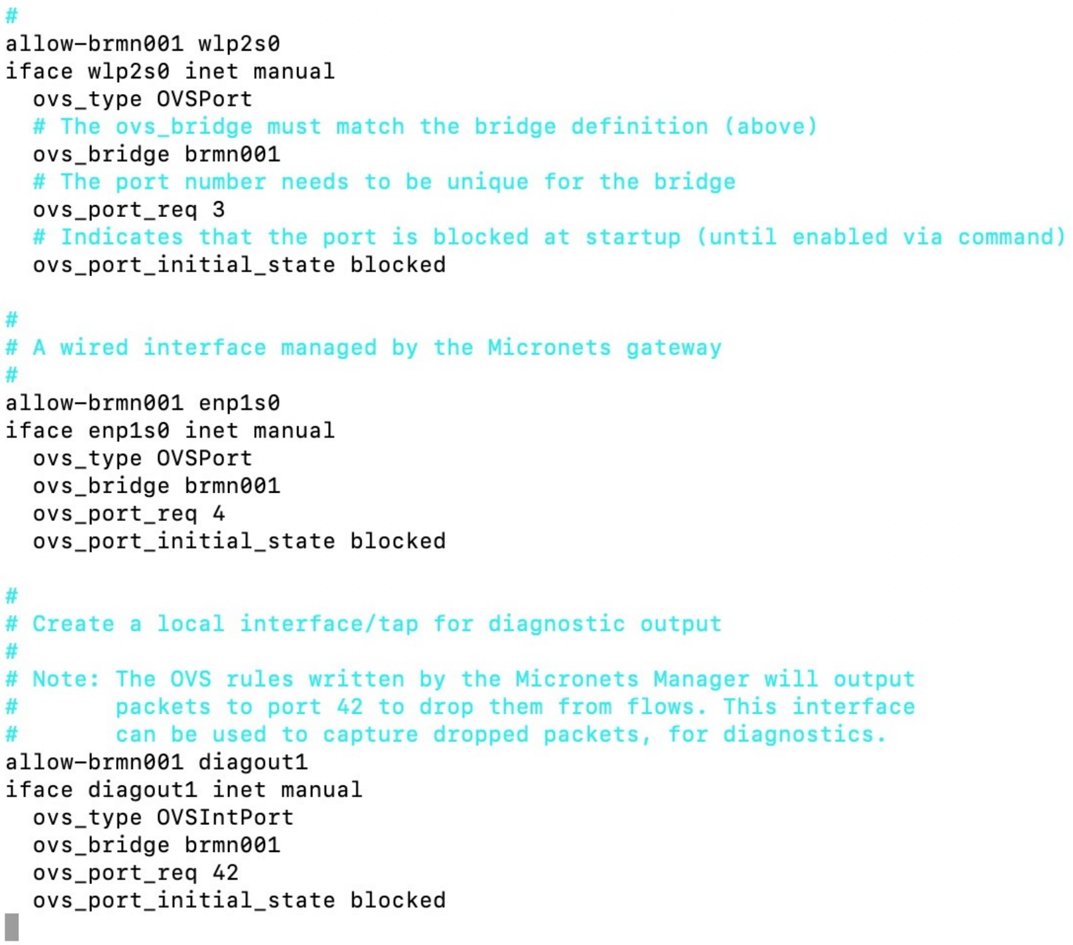
   b. Navigate to the /etc/network/interfaces file on the gateway, and copy the subnets configurations, which will be used for the gateway configuration file in the following steps:

      ```
      sudo vim /etc/network/interfaces
      ```

      Copy and save the subnet and ranges associated with the interfaces identified in the previous step from this file (Note: These are at the bottom of the file):



```
# Note: The entries below are sample definitions to be added to the
#       system-provided /etc/network/interfaces file. The definitions
#       include custom keywords to setup the OVS bridge and network
#       configuration.
auto enp0s31f6
iface enp0s31f6 inet static
  address 192.168.1.30/24
  gateway 192.168.1.1
  dns-nameservers 8.8.8.8 8.8.4.4
#
# create an OpenVswitch bridge for Micronets management
#
auto brmn001
allow-ovs brmn001
iface brmn001 inet manual
  ovs_type OVSBridge
  # This is the port that's connected to the Internet
  ovs_bridge_uplink_port enp0s31f6
  # the ovs_ports should list all wired and wireless interfaces under Micronets management
  ovs_ports diagout1 wlp2s0 enp1s0
  ovs_protocols OpenFlow10,OpenFlow11,OpenFlow12,OpenFlow13

# Assign IP addresses to the bridge that may be configured as Micronets
# Note: This will be replaced with dynamic route table entries in the future

iface brmn001 inet static
  address 10.135.1.1/24

iface brmn001 inet static
  address 10.135.2.1/24

iface brmn001 inet static
  address 10.135.3.1/24

iface brmn001 inet static
  address 10.135.4.1/24

iface brmn001 inet static
  address 10.135.5.1/24

#
# The uplink port
#

# An uplink may already be defined in the system-provided interfaces file.
# This interface should have a default gateway and must NOT be listed in the
# ovs_ports line of the bridge definition.

#
# A wireless interface managed by the Micronets gateway
```



```
#
allow-brmn001 wlp2s0
iface wlp2s0 inet manual
  ovs_type OVSPort
  # The ovs_bridge must match the bridge definition (above)
  ovs_bridge brmn001
  # The port number needs to be unique for the bridge
  ovs_port_req 3
  # Indicates that the port is blocked at startup (until enabled via command)
  ovs_port_initial_state blocked

#
# A wired interface managed by the Micronets gateway
#
allow-brmn001 enp1s0
iface enp1s0 inet manual
  ovs_type OVSPort
  ovs_bridge brmn001
  ovs_port_req 4
  ovs_port_initial_state blocked

#
# Create a local interface/tap for diagnostic output
#
# Note: The OVS rules written by the Micronets Manager will output
#       packets to port 42 to drop them from flows. This interface
#       can be used to capture dropped packets, for diagnostics.
allow-brmn001 diagout1
iface diagout1 inet manual
  ovs_type OVSIntPort
  ovs_bridge brmn001
  ovs_port_req 42
  ovs_port_initial_state blocked
```

c. Create the gateway config file by entering the following command:

```
sudo vim gateway-config-001.json
```

d. Modify the following configuration to include your gateway's MAC address and subnets as seen below and copy them into the *gateway-config-001.json* file:

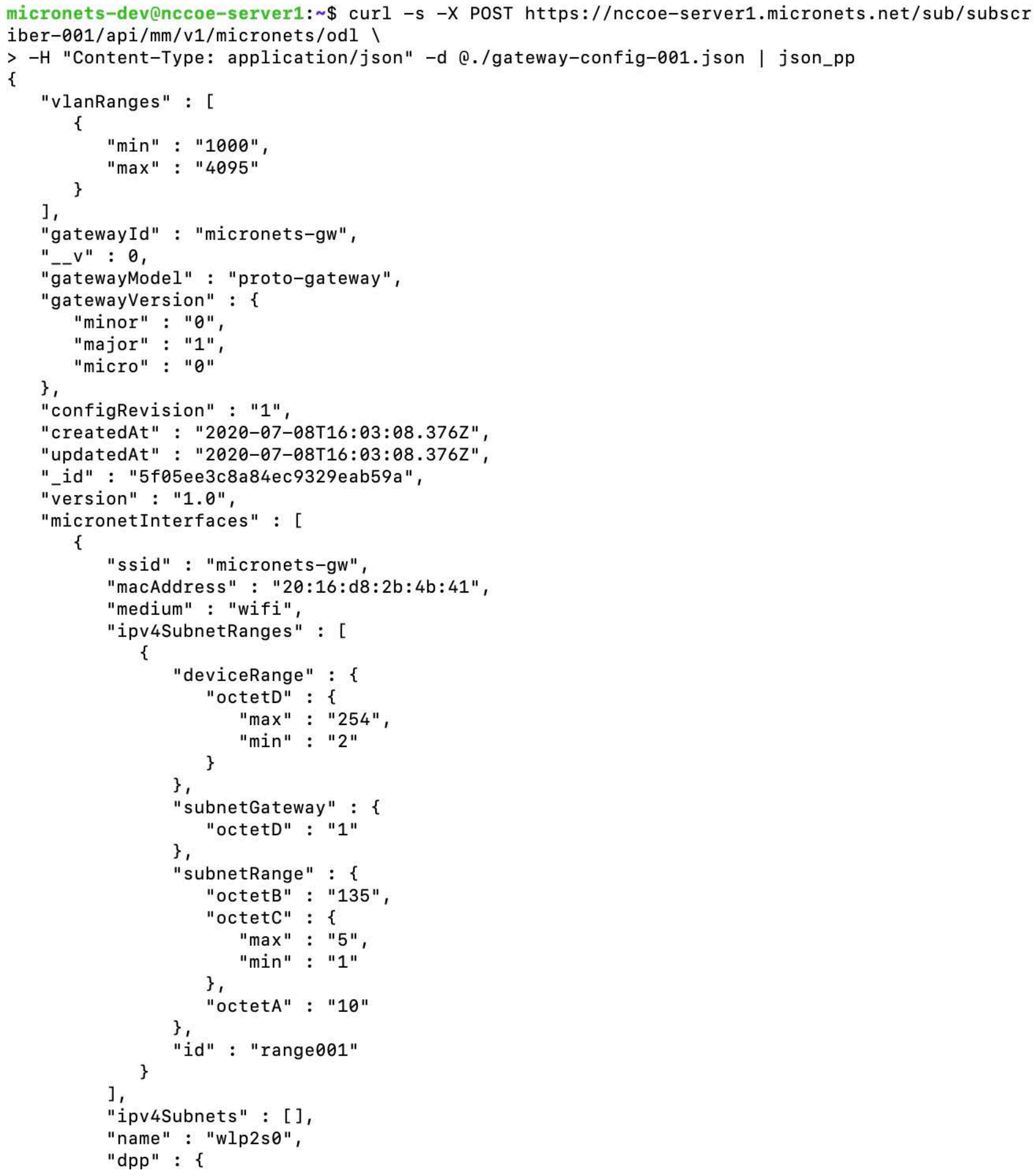
Be sure to modify the **ipv4SubnetRanges** definition to match the bridge subnet range—e.g., the file above defines five different subnets ranging from 10.135.1.1/24–10.135.5.1/24, so we set **octetC** to have a minimum of 1 and a maximum of 5 and **octetD** to have a minimum of 2 and a maximum of 254 as seen in the config below:

```
{
    "version": "1.0",
    "gatewayId": "micronets-gw",
    "gatewayModel": "proto-gateway",
    "gatewayVersion": {"major":1, "minor":0, "micro":0},
    "configRevision": 1,
    "vlanRanges": [
        {"min":1000, "max":4095}
    ],
    "micronetInterfaces": [
        {
            "medium": "wifi",
            "name": "wlp2s0",
            "macAddress": "20:16:d8:2b:4b:41",
            "ssid": "micronets-gw",
            "dpp": {
                "supportedAkms": ["psk"]
            },
            "ipv4SubnetRanges": [
                {
                    "id": "range001",
                    "subnetRange": {"octetA": 10,
                                  "octetB": 135,
                                  "octetC": {"min":1, "max":5}
                    },
                    "subnetGateway": {"octetD": 1},
                    "deviceRange": {"octetD": {"min":2, "max":254}}
                }
            ]
        },
        {
            "medium": "ethernet",
            "name": "enp1s0",
            "macAddress": "80:ee:73:dc:64:1d",
            "ipv4Subnets": [
                {
                    "id": "range001",
                    "subnetRange": {"octetA": 10,
                                  "octetB": 135,
                                  "octetC": 250
                    },
                    "subnetGateway": {"octetD": 1},
                    "deviceRange": {"octetD": {"min":2, "max":254}}
                }
            ]
        }
    ]
}
```

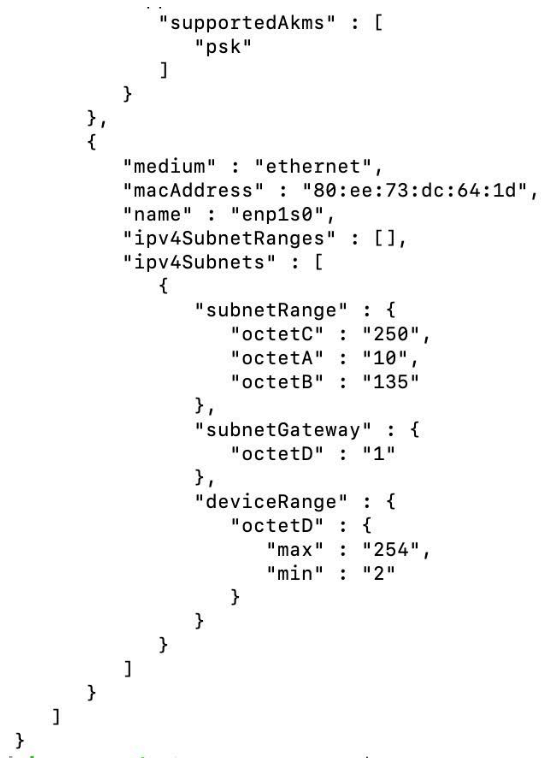
Register a gateway configuration for a subscriber with the subscriber's Micronets Manager instance by entering the following command (with the subscriber being subscriber-001 in this case):

```
curl -s -X POST https://nccoe-server1.micronets.net/sub/subscriber-
001/api/mm/v1/micronets/odl \
-H "Content-Type: application/json" -d @./gateway-config-001.json | json_pp
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -s -X POST https://nccoe-server1.micronets.net/sub/subscr
iber-001/api/mm/v1/micronets/odl \
> -H "Content-Type: application/json" -d @./gateway-config-001.json | json_pp
{
   "vlanRanges" : [
      {
         "min" : "1000",
         "max" : "4095"
      }
   ],
   "gatewayId" : "micronets-gw",
   "__v" : 0,
   "gatewayModel" : "proto-gateway",
   "gatewayVersion" : {
      "minor" : "0",
      "major" : "1",
      "micro" : "0"
   },
   "configRevision" : "1",
   "createdAt" : "2020-07-08T16:03:08.376Z",
   "updatedAt" : "2020-07-08T16:03:08.376Z",
   "_id" : "5f05ee3c8a84ec9329eab59a",
   "version" : "1.0",
   "micronetInterfaces" : [
      {
         "ssid" : "micronets-gw",
         "macAddress" : "20:16:d8:2b:4b:41",
         "medium" : "wifi",
         "ipv4SubnetRanges" : [
            {
               "deviceRange" : {
                  "octetD" : {
                     "max" : "254",
                     "min" : "2"
                  }
               },
               "subnetGateway" : {
                  "octetD" : "1"
               },
               "subnetRange" : {
                  "octetB" : "135",
                  "octetC" : {
                     "max" : "5",
                     "min" : "1"
                  },
                  "octetA" : "10"
               },
               "id" : "range001"
            }
         ],
         "ipv4Subnets" : [],
         "name" : "wlp2s0",
         "dpp" : {
```

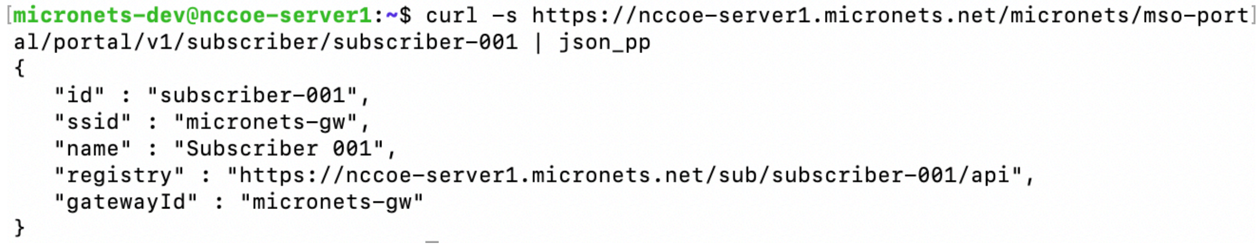


```
            "supportedAkms" : [
               "psk"
            ]
         }
      },
      {
         "medium" : "ethernet",
         "macAddress" : "80:ee:73:dc:64:1d",
         "name" : "enp1s0",
         "ipv4SubnetRanges" : [],
         "ipv4Subnets" : [
            {
               "subnetRange" : {
                  "octetC" : "250",
                  "octetA" : "10",
                  "octetB" : "135"
               },
               "subnetGateway" : {
                  "octetD" : "1"
               },
               "deviceRange" : {
                  "octetD" : {
                     "max" : "254",
                     "min" : "2"
                  }
               }
            }
         ]
      }
   ]
}
```

Confirm that the gateway ID is updated in the MSO portal by executing the following command:

```
curl -s https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/subscriber/subscriber-001 | json_pp
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -s https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/subscriber/subscriber-001 | json_pp
{
   "id" : "subscriber-001",
   "ssid" : "micronets-gw",
   "name" : "Subscriber 001",
   "registry" : "https://nccoe-server1.micronets.net/sub/subscriber-001/api",
   "gatewayId" : "micronets-gw"
}
```

Configure the Micronets Gateway with the Websocket Proxy keys provisioned for the gateway:

Copy the client cert and key as well as the Websocket root certificate, created in the product installation section, from the cloud server into the gateway by executing the following commands from the gateway:

i. Copy the *micronets-gw-service.pkeycert.pem* to the gateway:

```
scp micronets-dev@nccoe-server1.micronets.net:Projects/micronets/micronets-gw-service.pkeycert.pem .
```

You should see the following output:

```
micronets-gw-service.pkeycert.pem                                100%  933    15.4KB/s   00:00
```

ii. Copy the *micronets-ws-root.cert.pem* to the gateway:

```
scp micronets-dev@nccoe-server1.micronets.net:Projects/micronets/micronets-ws-root.cert.pem .
```

You should see the following output:

```
micronets-ws-root.cert.pem                                       100%  656    10.8KB/s   00:00
```

b. Copy them into the gateway service library to be loaded when the gateway is restarted:

```
sudo cp -v micronets-gw-service.pkeycert.pem micronets-ws-root.cert.pem
/opt/micronets-gw/lib/
```

Change the Websocket lookup URL to use the MSO portal service on your server by completing the following commands:

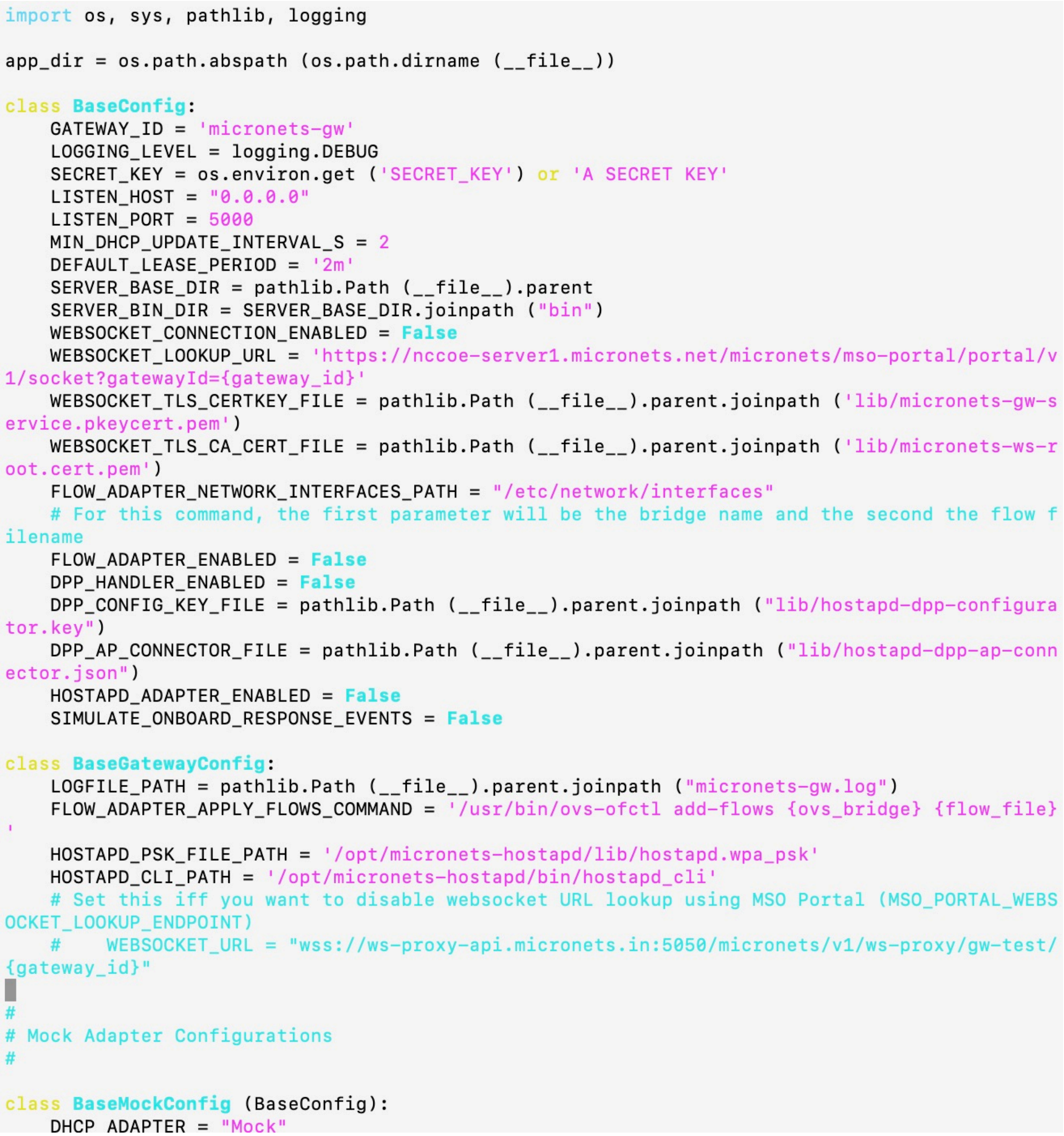
a. Open the Micronets Gateway config file by executing the following command:

```
sudo vim /opt/micronets-gw/config.py
```

b. Modify the **WEBSOCKET_LOOKUP_URL** and **GATEWAY_ID** to match the MSO portal Websocket lookup end point created in the product installation section and the Micronets Gateway ID:

```
WEBSOCKET_LOOKUP_URL = 'https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/socket?gatewayId={gateway_id}'
```

```
GATEWAY_ID = 'micronets-gw'
```



```
import os, sys, pathlib, logging

app_dir = os.path.abspath (os.path.dirname (__file__))

class BaseConfig:
    GATEWAY_ID = 'micronets-gw'
    LOGGING_LEVEL = logging.DEBUG
    SECRET_KEY = os.environ.get ('SECRET_KEY') or 'A SECRET KEY'
    LISTEN_HOST = "0.0.0.0"
    LISTEN_PORT = 5000
    MIN_DHCP_UPDATE_INTERVAL_S = 2
    DEFAULT_LEASE_PERIOD = '2m'
    SERVER_BASE_DIR = pathlib.Path (__file__).parent
    SERVER_BIN_DIR = SERVER_BASE_DIR.joinpath ("bin")
    WEBSOCKET_CONNECTION_ENABLED = False
    WEBSOCKET_LOOKUP_URL = 'https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/socket?gatewayId={gateway_id}'
    WEBSOCKET_TLS_CERTKEY_FILE = pathlib.Path (__file__).parent.joinpath ('lib/micronets-gw-service.pkeycert.pem')
    WEBSOCKET_TLS_CA_CERT_FILE = pathlib.Path (__file__).parent.joinpath ('lib/micronets-ws-root.cert.pem')
    FLOW_ADAPTER_NETWORK_INTERFACES_PATH = "/etc/network/interfaces"
    # For this command, the first parameter will be the bridge name and the second the flow filename
    FLOW_ADAPTER_ENABLED = False
    DPP_HANDLER_ENABLED = False
    DPP_CONFIG_KEY_FILE = pathlib.Path (__file__).parent.joinpath ("lib/hostapd-dpp-configurator.key")
    DPP_AP_CONNECTOR_FILE = pathlib.Path (__file__).parent.joinpath ("lib/hostapd-dpp-ap-connector.json")
    HOSTAPD_ADAPTER_ENABLED = False
    SIMULATE_ONBOARD_RESPONSE_EVENTS = False

class BaseGatewayConfig:
    LOGFILE_PATH = pathlib.Path (__file__).parent.joinpath ("micronets-gw.log")
    FLOW_ADAPTER_APPLY_FLOWS_COMMAND = '/usr/bin/ovs-ofctl add-flows {ovs_bridge} {flow_file}'

    HOSTAPD_PSK_FILE_PATH = '/opt/micronets-hostapd/lib/hostapd.wpa_psk'
    HOSTAPD_CLI_PATH = '/opt/micronets-hostapd/bin/hostapd_cli'
    # Set this iff you want to disable websocket URL lookup using MSO Portal (MSO_PORTAL_WEBSOCKET_LOOKUP_ENDPOINT)
    #    WEBSOCKET_URL = "wss://ws-proxy-api.micronets.in:5050/micronets/v1/ws-proxy/gw-test/{gateway_id}"

#
# Mock Adapter Configurations
#

class BaseMockConfig (BaseConfig):
    DHCP_ADAPTER = "Mock"
```

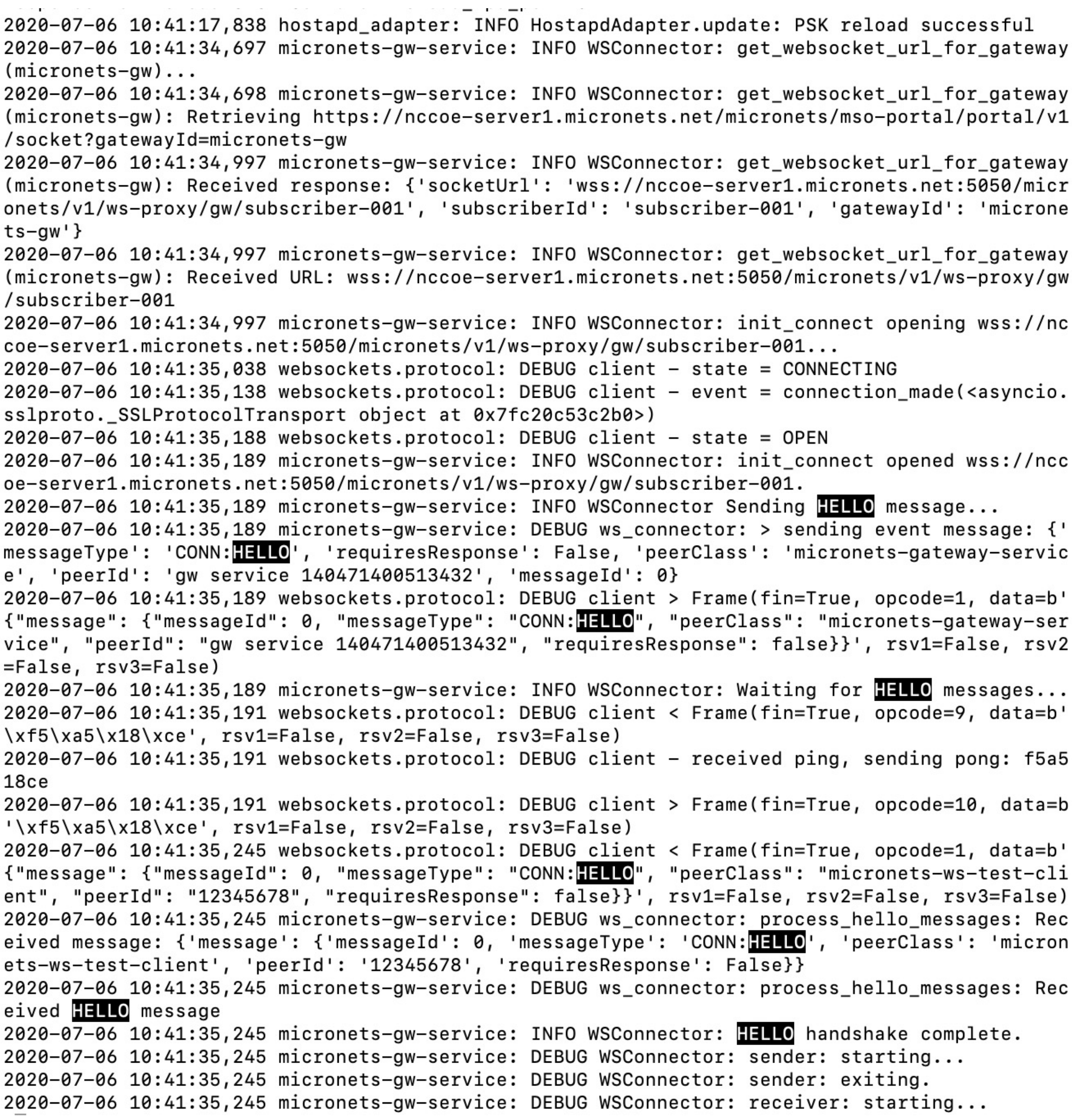
Restart the Micronets Gateway Service by executing the following command:

```
sudo systemctl restart micronets-gw.service
```

Check the Micronets Gateway Service log (**/opt/micronets-gw/micronets-gw.log**) to verify that the gateway's Websocket registration status was successful:

```
cat /opt/micronets-gw/micronets-gw.log
```

You should see output similar to the following:

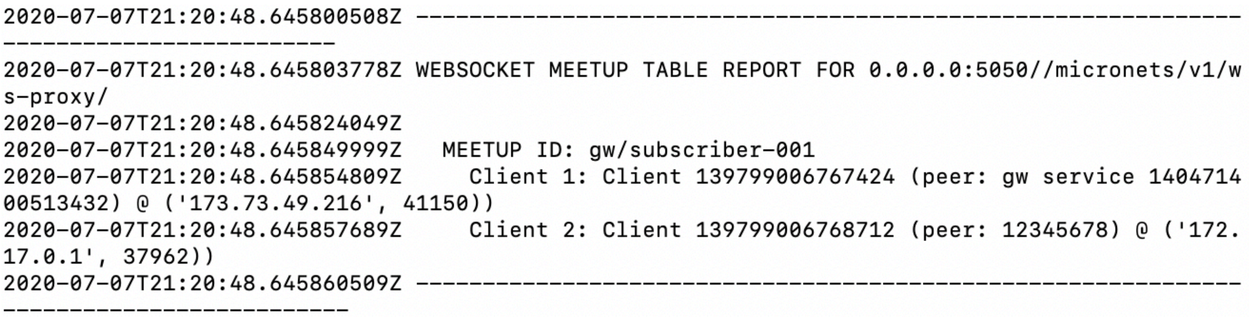


```
2020-07-06 10:41:17,838 hostapd_adapter: INFO HostapdAdapter.update: PSK reload successful
2020-07-06 10:41:34,697 micronets-gw-service: INFO WSConnector: get_websocket_url_for_gateway
(micronets-gw)...
2020-07-06 10:41:34,698 micronets-gw-service: INFO WSConnector: get_websocket_url_for_gateway
(micronets-gw): Retrieving https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1
/socket?gatewayId=micronets-gw
2020-07-06 10:41:34,997 micronets-gw-service: INFO WSConnector: get_websocket_url_for_gateway
(micronets-gw): Received response: {'socketUrl': 'wss://nccoe-server1.micronets.net:5050/micr
onets/v1/ws-proxy/gw/subscriber-001', 'subscriberId': 'subscriber-001', 'gatewayId': 'microne
ts-gw'}
2020-07-06 10:41:34,997 micronets-gw-service: INFO WSConnector: get_websocket_url_for_gateway
(micronets-gw): Received URL: wss://nccoe-server1.micronets.net:5050/micronets/v1/ws-proxy/gw
/subscriber-001
2020-07-06 10:41:34,997 micronets-gw-service: INFO WSConnector: init_connect opening wss://nc
coe-server1.micronets.net:5050/micronets/v1/ws-proxy/gw/subscriber-001...
2020-07-06 10:41:35,038 websockets.protocol: DEBUG client - state = CONNECTING
2020-07-06 10:41:35,138 websockets.protocol: DEBUG client - event = connection_made(<asyncio.
sslproto._SSLProtocolTransport object at 0x7fc20c53c2b0>)
2020-07-06 10:41:35,188 websockets.protocol: DEBUG client - state = OPEN
2020-07-06 10:41:35,189 micronets-gw-service: INFO WSConnector: init_connect opened wss://ncc
oe-server1.micronets.net:5050/micronets/v1/ws-proxy/gw/subscriber-001.
2020-07-06 10:41:35,189 micronets-gw-service: INFO WSConnector Sending HELLO message...
2020-07-06 10:41:35,189 micronets-gw-service: DEBUG ws_connector: > sending event message: {'
messageType': 'CONN:HELLO', 'requiresResponse': False, 'peerClass': 'micronets-gateway-servic
e', 'peerId': 'gw service 140471400513432', 'messageId': 0}
2020-07-06 10:41:35,189 websockets.protocol: DEBUG client > Frame(fin=True, opcode=1, data=b'
{"message": {"messageId": 0, "messageType": "CONN:HELLO", "peerClass": "micronets-gateway-ser
vice", "peerId": "gw service 140471400513432", "requiresResponse": false}}', rsv1=False, rsv2
=False, rsv3=False)
2020-07-06 10:41:35,189 micronets-gw-service: INFO WSConnector: Waiting for HELLO messages...
2020-07-06 10:41:35,191 websockets.protocol: DEBUG client < Frame(fin=True, opcode=9, data=b'
\xf5\xa5\x18\xce', rsv1=False, rsv2=False, rsv3=False)
2020-07-06 10:41:35,191 websockets.protocol: DEBUG client - received ping, sending pong: f5a5
18ce
2020-07-06 10:41:35,191 websockets.protocol: DEBUG client > Frame(fin=True, opcode=10, data=b
'\xf5\xa5\x18\xce', rsv1=False, rsv2=False, rsv3=False)
2020-07-06 10:41:35,245 websockets.protocol: DEBUG client < Frame(fin=True, opcode=1, data=b'
{"message": {"messageId": 0, "messageType": "CONN:HELLO", "peerClass": "micronets-ws-test-cli
ent", "peerId": "12345678", "requiresResponse": false}}', rsv1=False, rsv2=False, rsv3=False)
2020-07-06 10:41:35,245 micronets-gw-service: DEBUG ws_connector: process_hello_messages: Rec
eived message: {'message': {'messageId': 0, 'messageType': 'CONN:HELLO', 'peerClass': 'micron
ets-ws-test-client', 'peerId': '12345678', 'requiresResponse': False}}
2020-07-06 10:41:35,245 micronets-gw-service: DEBUG ws_connector: process_hello_messages: Rec
eived HELLO message
2020-07-06 10:41:35,245 micronets-gw-service: INFO WSConnector: HELLO handshake complete.
2020-07-06 10:41:35,245 micronets-gw-service: DEBUG WSConnector: sender: starting...
2020-07-06 10:41:35,245 micronets-gw-service: DEBUG WSConnector: sender: exiting.
2020-07-06 10:41:35,245 micronets-gw-service: DEBUG WSConnector: receiver: starting...
```

Confirm the establishment of the gateway-manager control connection by examining the Websocket Proxy connection reports in the Websocket Proxy log:

```
/etc/micronets/micronets-ws-proxy docker-logs | less
```

Look for the following in the log (with the **MEETUP ID** matching the subscriber name in question):



```
2020-07-07T21:20:48.645800508Z -------------------------------------------------------------
-------------------------
2020-07-07T21:20:48.645803778Z WEBSOCKET MEETUP TABLE REPORT FOR 0.0.0.0:5050//micronets/v1/w
s-proxy/
2020-07-07T21:20:48.645824049Z
2020-07-07T21:20:48.645849999Z   MEETUP ID: gw/subscriber-001
2020-07-07T21:20:48.645854809Z     Client 1: Client 139799006767424 (peer: gw service 1404714
00513432) @ ('173.73.49.216', 41150))
2020-07-07T21:20:48.645857689Z     Client 2: Client 139799006768712 (peer: 12345678) @ ('172.
17.0.1', 37962))
2020-07-07T21:20:48.645860509Z -------------------------------------------------------------
--------------------------
```

This indicates that the Micronets Gateway Service and the Micronets Manager for the subscriber connected and can exchange provisioning commands and event indications.

4.2.3 Integrating Micronets Proto-Pi Device

This section describes associating an MSO portal subscriber with the Micronets Gateway. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation: https://github.com/cablelabs/micronets-pi3/blob/nccoe-build-3/README.md#Operation.

4.2.3.1 Prerequisites

To successfully complete this section, be sure to have completed the product installation section associated with the Micronets Proto-Pi device. Ensure all steps have been successfully completed before proceeding to the instructions.

4.2.3.2 Instructions

1. Connect to the Raspberry Pi via SSH by entering the following command:

   ```
   ssh pi@192.168.30.191
   ```

   You will be prompted to enter the device password the password will remain the same.

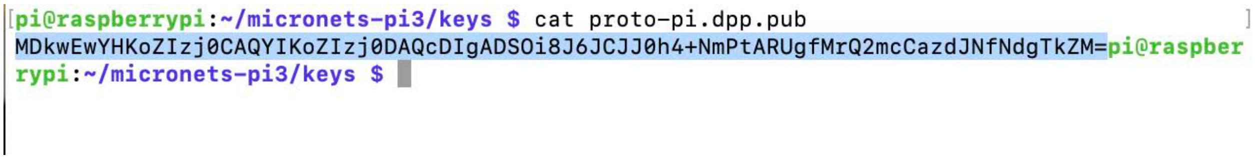
2. Change to the keys directory by entering the following command:

   ```
   cd micronets-pi3/keys/
   ```

3. Output the content of the **proto-pi.dpp.pub** file to copy the public key for this device (Note: You will need to store this device key for registering the device with the MUD registry if doing so manually):

```
cat proto-pi.dpp.pub
```

Highlight and copy the key that was output by the previous command:

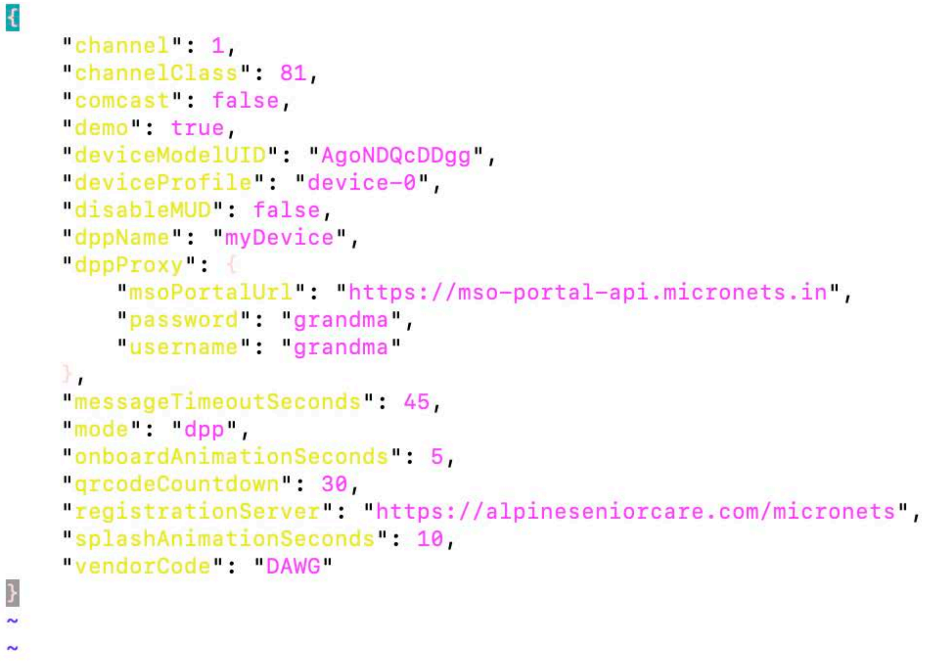


```
pi@raspberrypi:~/micronets-pi3/keys $ cat proto-pi.dpp.pub
MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTkZM=pi@raspber
rypi:~/micronets-pi3/keys $
```

4. Modify the **config.json** file to include the key that was copied in the previous step, and modify the parameters of the file to match your setup:

```
sudo vim ~/micronets-pi3/config/config.json
```

The original file before editing should be similar to the following screenshot:



```
{
    "channel": 1,
    "channelClass": 81,
    "comcast": false,
    "demo": true,
    "deviceModelUID": "AgoNDQcDDgg",
    "deviceProfile": "device-0",
    "disableMUD": false,
    "dppName": "myDevice",
    "dppProxy": {
        "msoPortalUrl": "https://mso-portal-api.micronets.in",
        "password": "grandma",
        "username": "grandma"
    },
    "messageTimeoutSeconds": 45,
    "mode": "dpp",
    "onboardAnimationSeconds": 5,
    "qrcodeCountdown": 30,
    "registrationServer": "https://alpineseniorcare.com/micronets",
    "splashAnimationSeconds": 10,
    "vendorCode": "DAWG"
}
~
~
```

If doing manual device registration edit the file to reflect the correct **DeviceModelUID** (should be the same name as the MUD file associated with this device), **dppMUDUrl**, **msoPortalUrl**, **registrationServer**, **vendorCode** as seen below:

```
{
    "channel": 1,
    "channelClass": 81,
    "comcast": false,
    "demo": true,
    "deviceModelUID": "nist-model-fe_northsouth.json",
    "deviceProfile": "device-0",
    "disableMUD": false,
    "dppMUDUrl": "https://nccoe-server1.microents.net/mud/v1/mud-
url/TEST/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgACxjMF8Ucp6d3gRBImv78eGEMwB5igS2Kt5b
nXI7VeBrc=",
    "dppName": "myDevice",
    "dppProxy": {
        "msoPortalUrl": "https://nccoe-server1.micronets.net/micronets/mso-por-
tal/",
        "password": "grandma",
        "username": "grandma"
    },
    "messageTimeoutSeconds": 45,
    "mode": "dpp",
    "onboardAnimationSeconds": 5,
    "qrcodeCountdown": 30,
    "registrationServer": "https://nccoe-server1.micronets.net/registry/de-
vices",
    "splashAnimationSeconds": 10,
    "vendorCode": "TEST"
}
```

If enabling self-registry, follow the steps described in the following documentation: https://github.com/cablelabs/micronets-pi3/blob/nccoe-build-3/README.md#dpp-mode-mud-registry .

5. Reboot the device for the new config file to take effect:

   ```
   sudo reboot
   ```

4.2.4 Updating MUD Registry

This section describes the HTTP API operations for interacting with the MUD registry. The instructions detail how to register a MUD-capable device and its MUD URL with a vendor. For additional API operations not documented here, follow the link to the CableLabs MUD registry operation documentation: https://github.com/cablelabs/micronets-mud-registry/blob/nccoe-build-3/README.md#Operation.

4.2.4.1 Prerequisites

To successfully complete this section, be sure to have completed the product installation section.

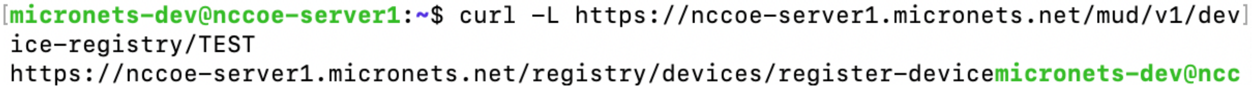
4.2.4.2 Instructions

1. Retrieve the device registry URL for a vendor by entering the following curl command:

/mud/v1/device-registry/:vendor-code

```
curl -L https://nccoe-server1.micronets.net/mud/v1/device-registry/TEST
```

You should see output similar to the following:



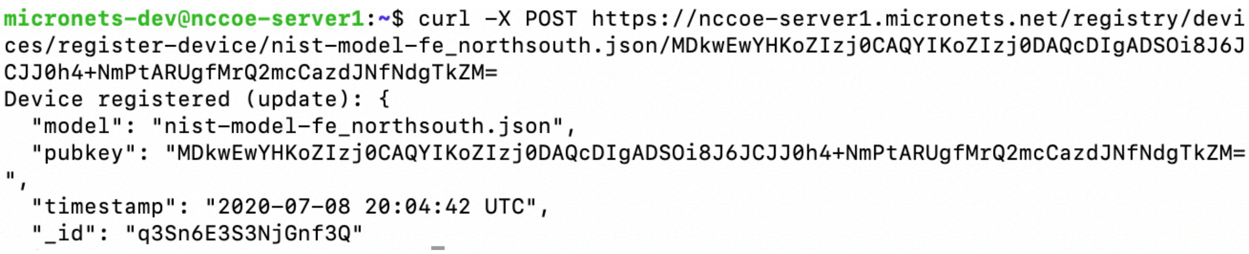
```
micronets-dev@nccoe-server1:~$ curl -L https://nccoe-server1.micronets.net/mud/v1/dev
ice-registry/TEST
https://nccoe-server1.micronets.net/registry/devices/register-devicemicronets-dev@ncc
```

2. Register a device with a vendor's registry. This requires the device model UID and the public key, which can be modified and retrieved through the Micronets Proto-Pi:

   /registry/devices/register-device/:device-model-UID64/:public-key

   ```
   curl -X POST https://nccoe-server1.micronets.net/registry/devices/register-
   device/nist-model-
   fe_northsouth.json/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfM
   rQ2mcCazdJNfNdgTkZM=
   ```

   You should see output similar to the following:



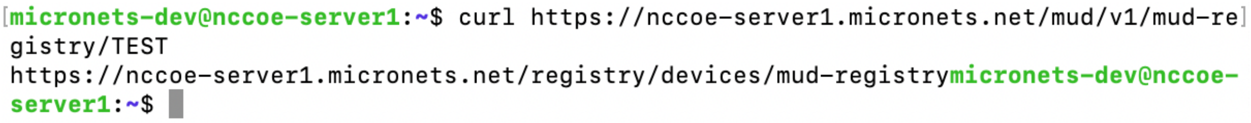
```
micronets-dev@nccoe-server1:~$ curl -X POST https://nccoe-server1.micronets.net/registry/devi
ces/register-device/nist-model-fe_northsouth.json/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6J
CJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTkZM=
Device registered (update): {
  "model": "nist-model-fe_northsouth.json",
  "pubkey": "MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTkZM=
",
  "timestamp": "2020-07-08 20:04:42 UTC",
  "_id": "q3Sn6E3S3NjGnf3Q"
```

Retrieve the MUD registry URL for a vendor:

/mud/v1/mud-registry/:vendor-code

```
curl https://nccoe-server1.micronets.net/mud/v1/mud-registry/TEST
```

You should see output similar to the following:



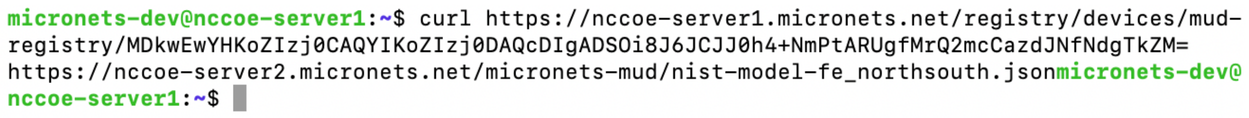
```
micronets-dev@nccoe-server1:~$ curl https://nccoe-server1.micronets.net/mud/v1/mud-re
gistry/TEST
https://nccoe-server1.micronets.net/registry/devices/mud-registrymicronets-dev@nccoe-
server1:~$ 
```

Lookup a MUD URL from the vendor MUD registry:

/registry/devices/mud-registry/:public-key

```
curl https://nccoe-server1.micronets.net/registry/devices/mud-registry/
MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTkZM
=
```

You should see output similar to the following:



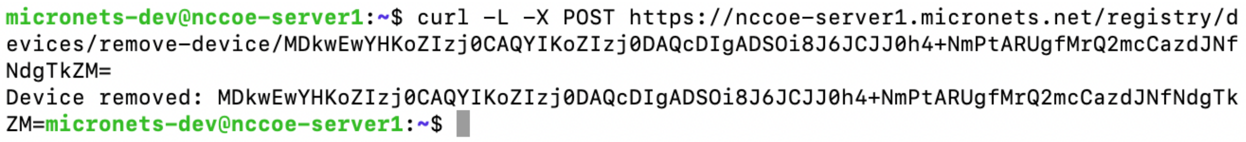
```
micronets-dev@nccoe-server1:~$ curl https://nccoe-server1.micronets.net/registry/devices/mud-
registry/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTkZM=
https://nccoe-server2.micronets.net/micronets-mud/nist-model-fe_northsouth.jsonmicronets-dev@
nccoe-server1:~$
```

Delete a device from the MUD registry (Note: If you do this step, the device will no longer be associated with a MUD file. Therefore, you should execute this command only if you do not intend to onboard the device with MUD capabilities):

/registry/devices/remove-device/:public-key

```
curl -L -X POST https://nccoe-server1.micronets.net/registry/devices/remove-de-
vice/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNd
gTkZM=
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -L -X POST https://nccoe-server1.micronets.net/registry/d
evices/remove-device/MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNf
NdgTkZM=
Device removed: MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADSOi8J6JCJJ0h4+NmPtARUgfMrQ2mcCazdJNfNdgTk
ZM=micronets-dev@nccoe-server1:~$
```

4.2.5 Integrating the Micronets iPhone App with MSO Portal

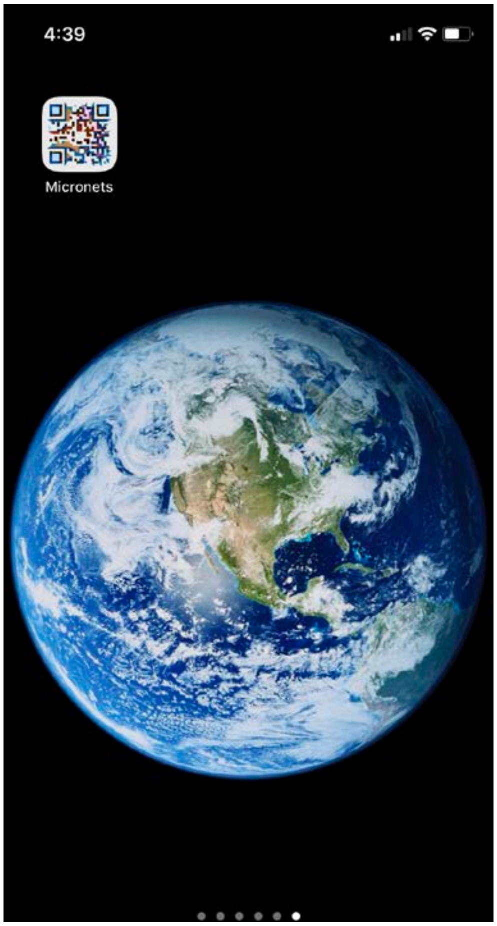
This section describes integrating the Micronets iPhone application with the MSO portal. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation: https://github.com/cablelabs/micronets-mobile/blob/nccoe-build-3/README.md#Operation.

4.2.5.1 Prerequisites

A valid network connection on the iPhone is required as well as the completion of the product installation section related to the Micronets iPhone application.

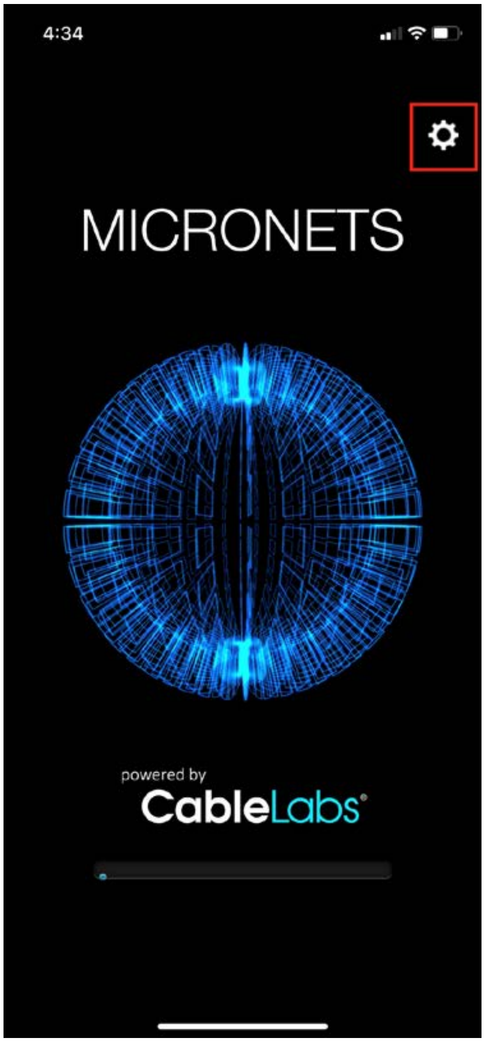
4.2.5.2 Instructions

1. Open the Micronets mobile application:



3141

3142 2. From the splash screen click the gear button in the upper right corner to open the settings page:

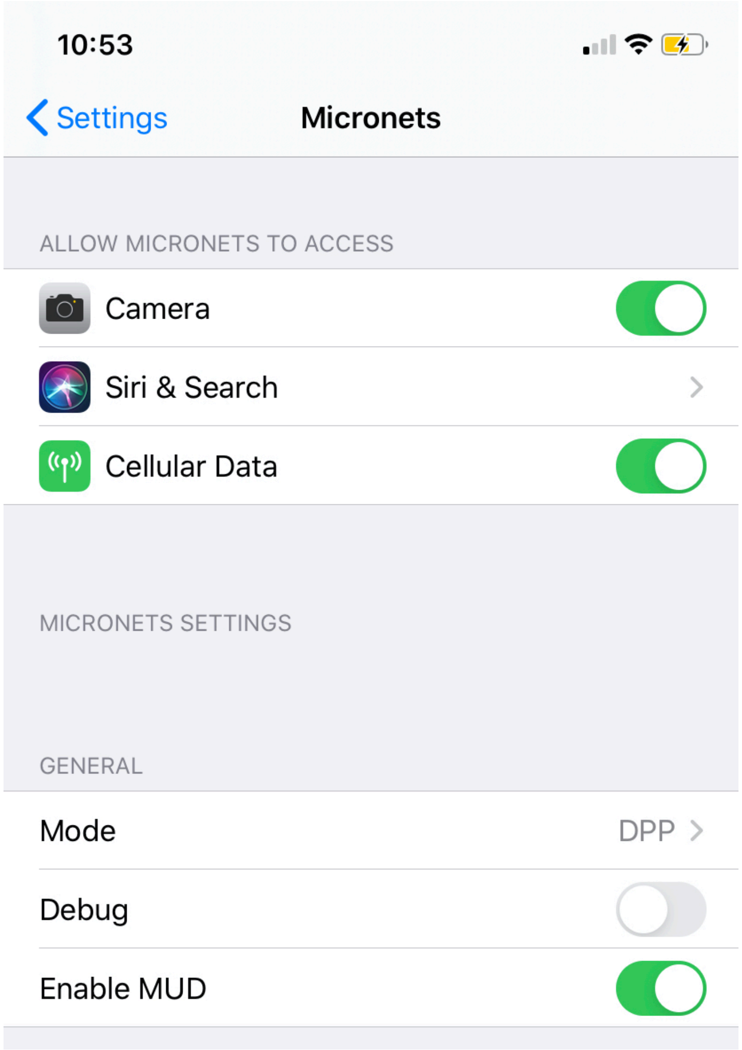


Modify the following fields in the general settings:

Mode - DPP or Clinic: We select DPP, if you are selecting the Clinic mode please follow the documentation for details related to the Clinic mode

Debug - Leave this off as CableLabs will be deprecating this in the future

Enable MUD – If enabled, it will try to fetch the MUD file for the scanned device and pre-populate the Submit form prior to onboarding.



10:53

< Settings

Micronets

ALLOW MICRONETS TO ACCESS

Camera ●

Siri & Search >

Cellular Data ●

MICRONETS SETTINGS

GENERAL

Mode DPP >

Debug ○

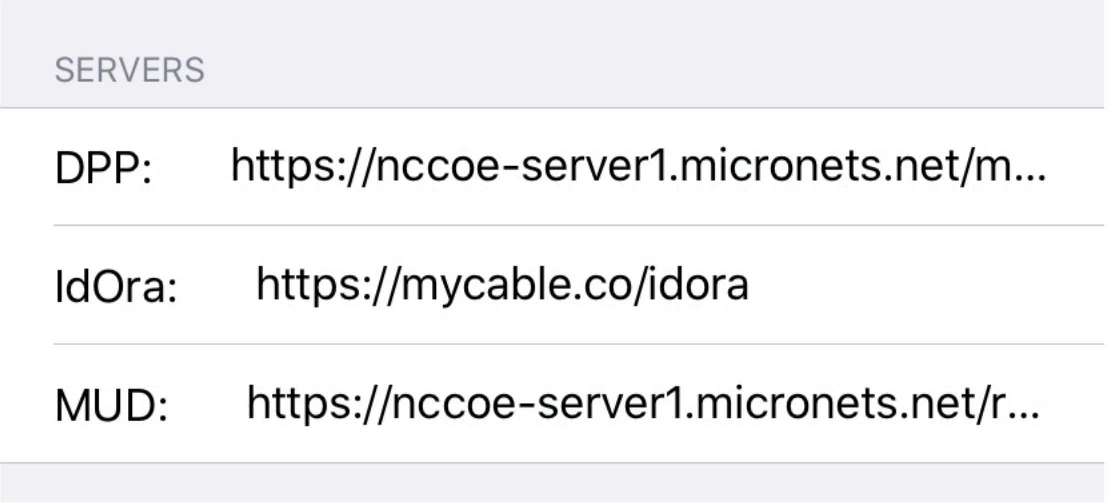
Enable MUD ●

Modify the servers for the Micronets application:

DPP – MSO portal server URL for submitting onboard requests

IdOra – Server for user authentication (Note: this is only required if utilizing the Clinic Mode)

MUD – MUD registry server for looking up MUD files using the vendor code and public key in the QRCode. (Note: this only needs to be changed if you are deploying your own MUD registry)



SERVERS

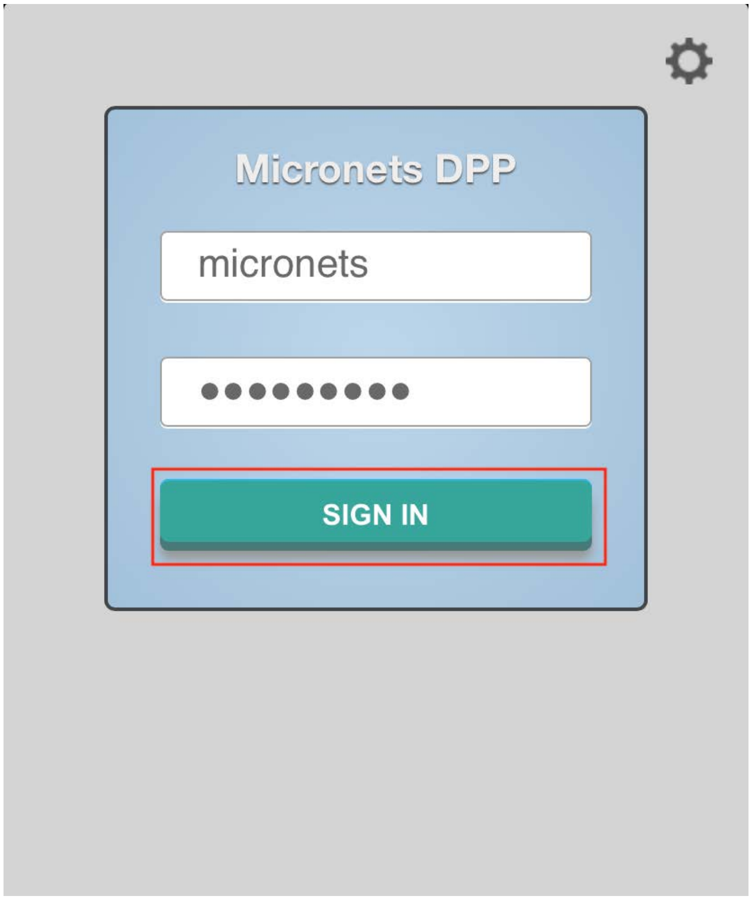
DPP: https://nccoe-server1.micronets.net/m...

IdOra: https://mycable.co/idora

MUD: https://nccoe-server1.micronets.net/r...

3157

3158 Back on the Micronets mobile application, enter your subscriber credentials and click **SIGN IN**:



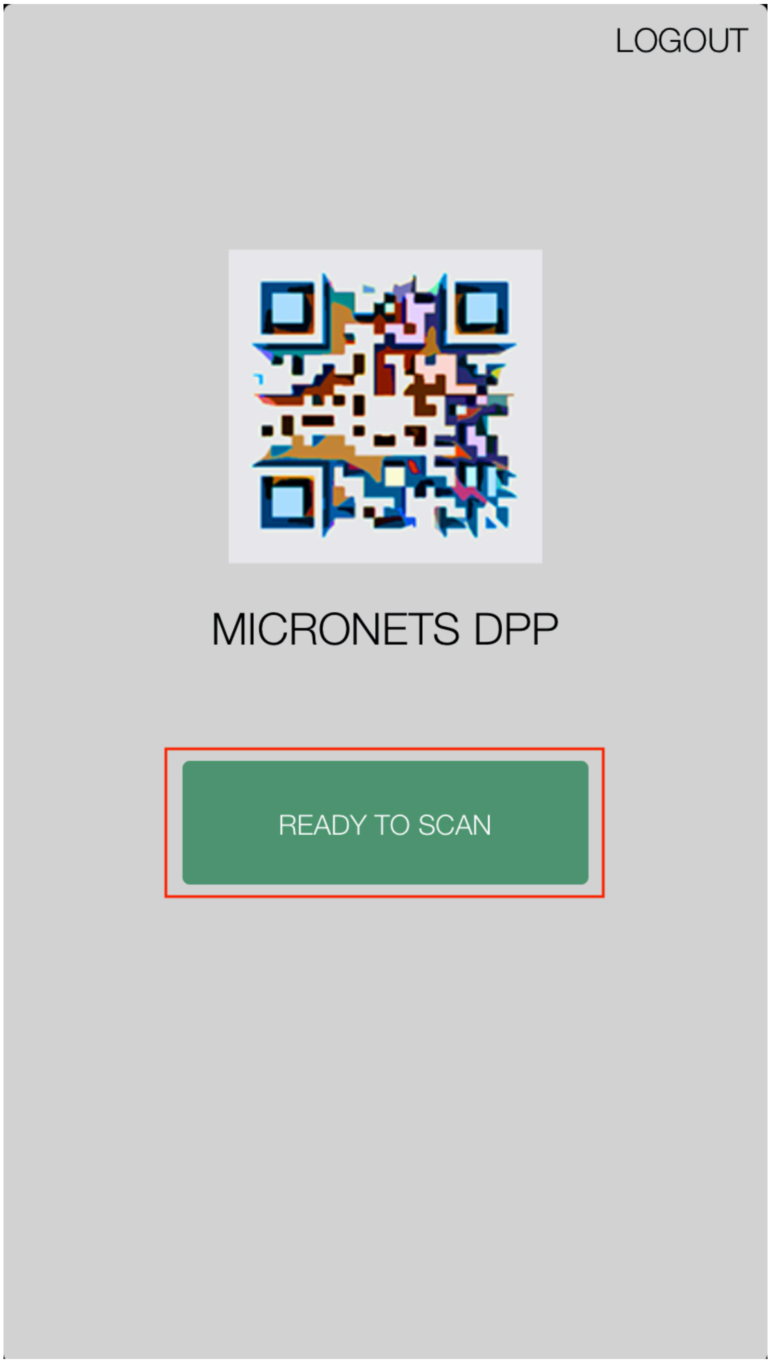
Micronets DPP

micronets

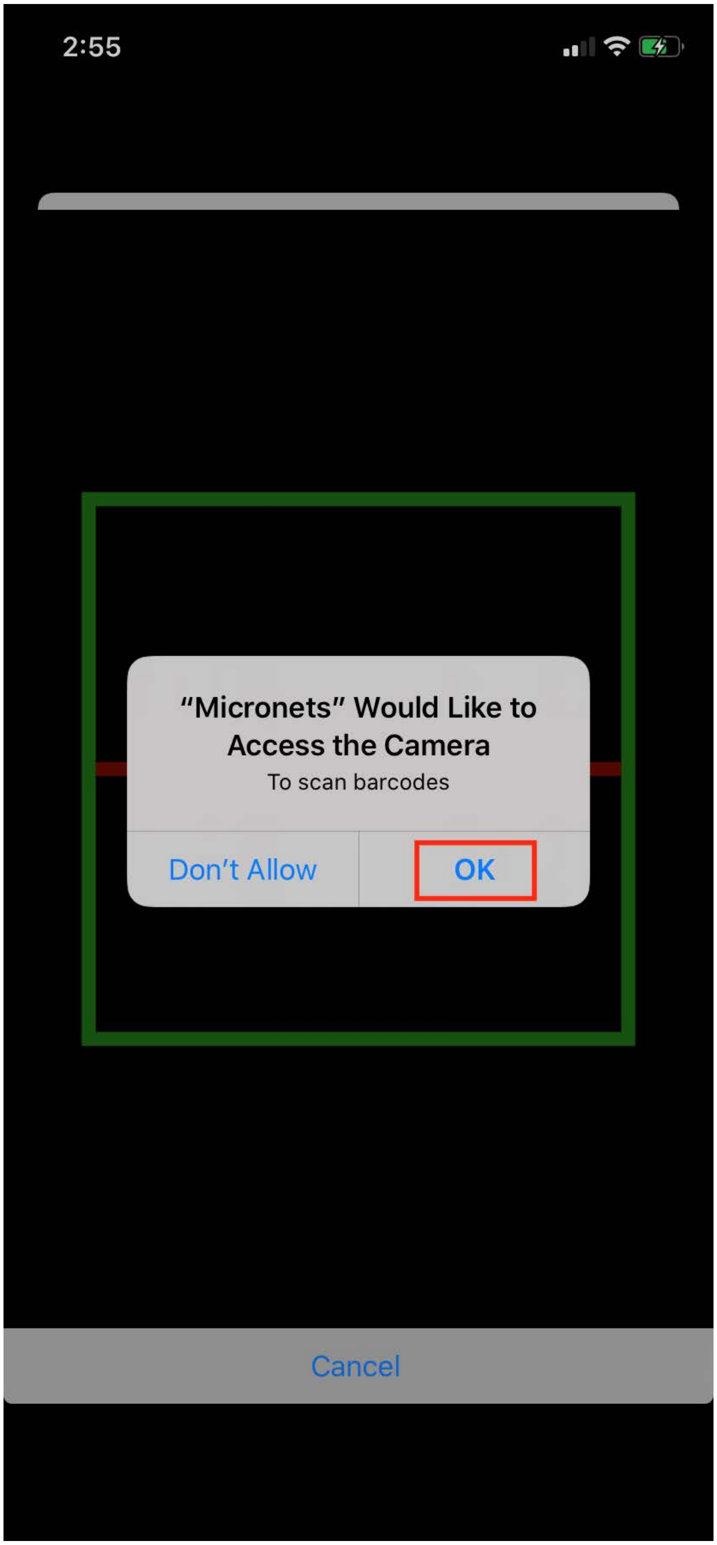
●●●●●●●●●

3159

Click the **READY TO SCAN** button to open the camera for onboarding:



If prompted, allow the Micronets application camera access, by clicking **OK**:



"Micronets" Would Like to Access the Camera

To scan barcodes

3163

4.2.6 Onboarding Micronets Proto-Pi to a micronet

This section describes how to onboard a configured Micronets Proto-Pi device to a micronet using the Micronet iPhone app. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation:https://github.com/cablelabs/micronets-pi3/blob/nccoe-build-3/README.md#Operation.

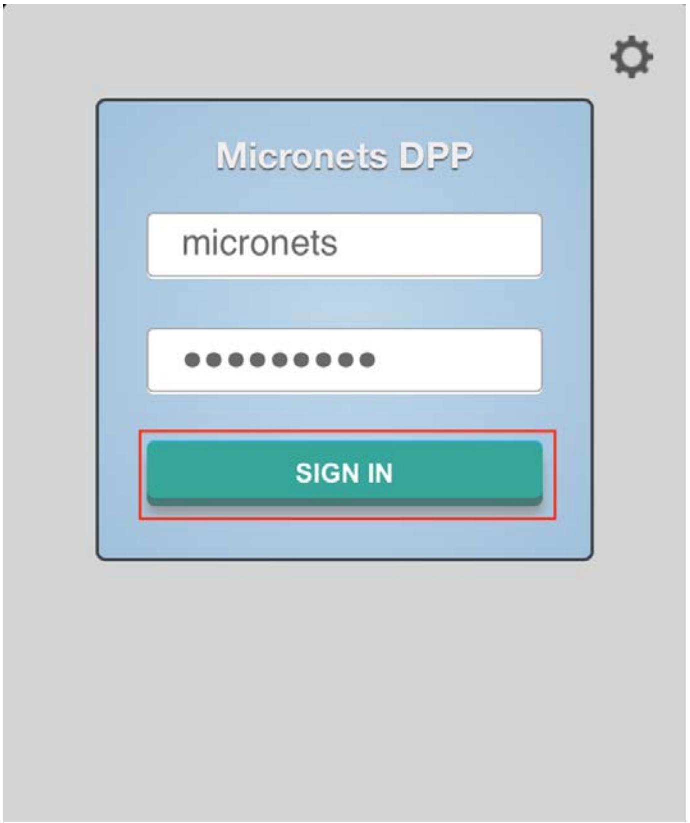
4.2.6.1 Prerequisites

To successfully complete this section the following is required:

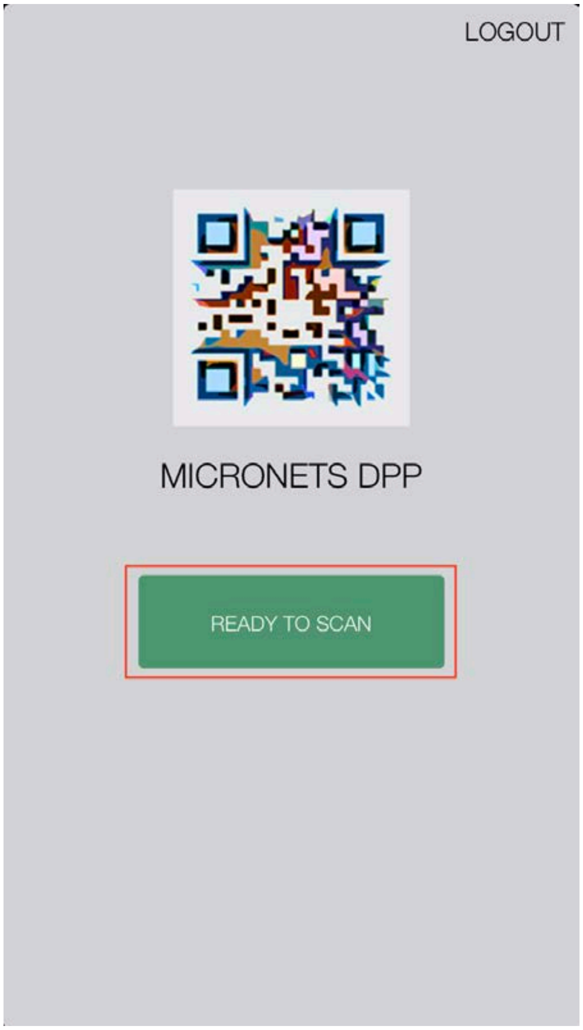
- a Raspberry Pi with the Micronets Proto-Pi software installed and configured
- an iOS or Android phone with the Micronets application installed and configured
- a Micronets subscriber account configured in Section 4.2.1
- a gateway device associated with the Micronets subscriber configured in Section 4.2.2

4.2.6.2 Instructions

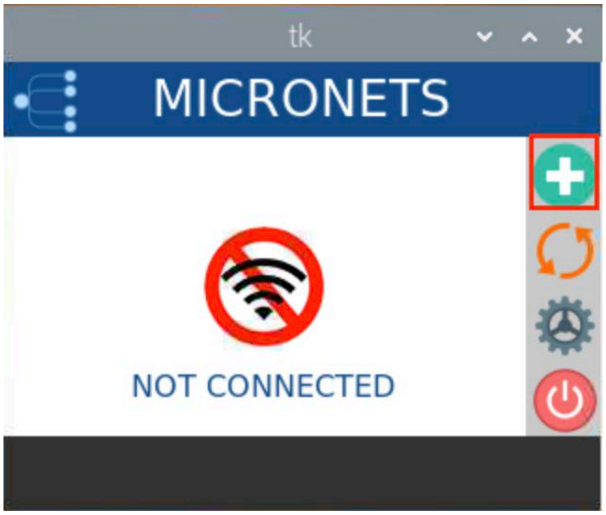
1. If leveraging the self-registration feature for MUD onboarding, ensure that an ethernet cable is connected to the Raspberry Pi running the Micronets Proto-Pi software.
2. Power on the Pi device. If leveraging the self-registration feature, the device will automatically be registered on first run.
3. On the mobile device, open the Micronets mobile application and log in with your subscriber credentials.



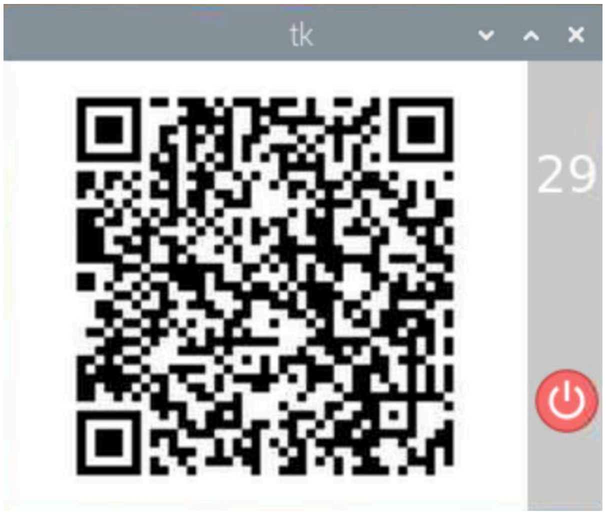
4. On the Mobile device, tap the **Ready to Scan** button:



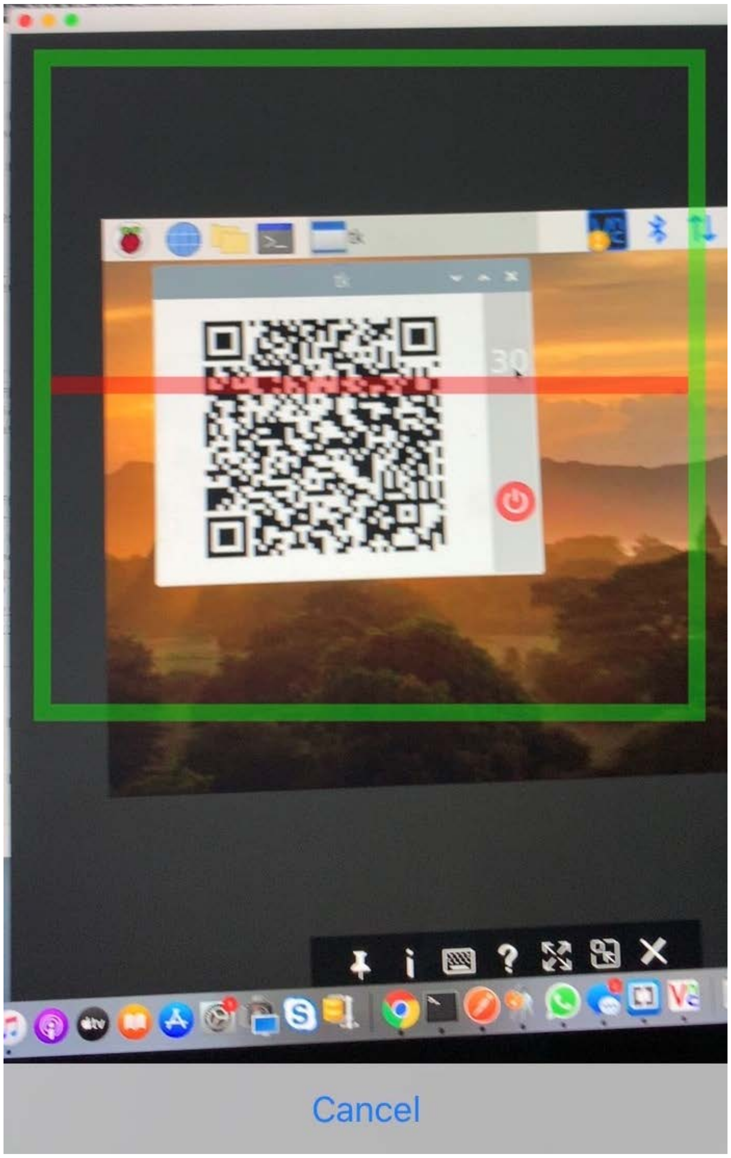
5. On the Pi, click the Onboard icon:



You should see a QR code appear on the screen:



6. Scan the QRCode with the Micronet mobile application:

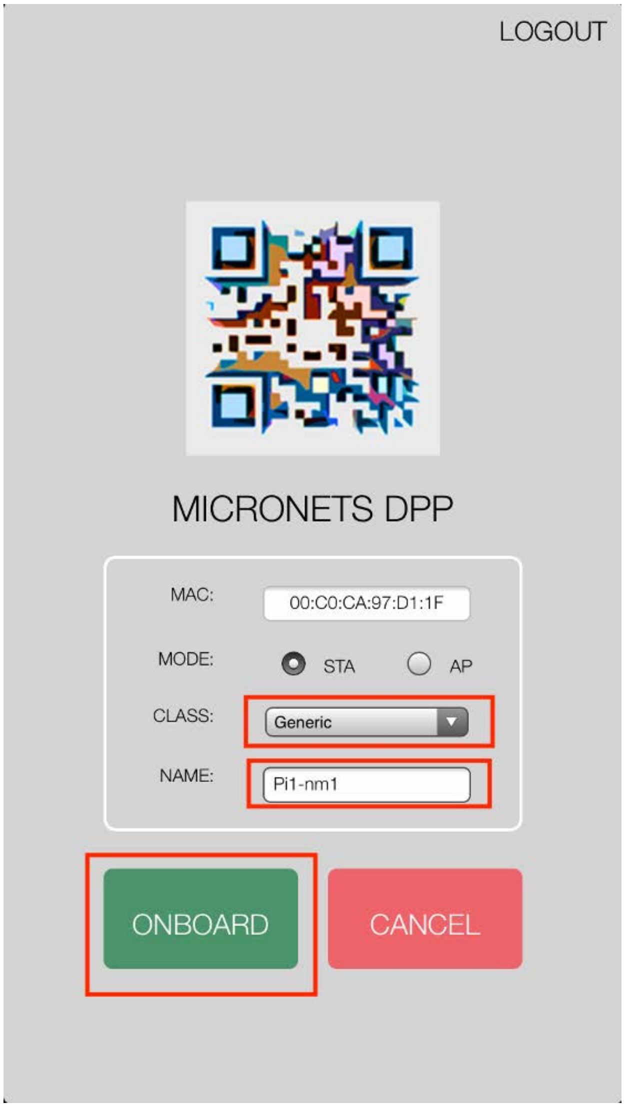


7. On the next screen that appears on the Micronets mobile application, input the following information in a timely fashion (Note: these steps must be completed while the device is still in onboard mode).

   a. If a MUD file was found, the device CLASS and NAME will be prepopulated, modify as needed. In the case that a MUD file was not found populate the **CLASS** and **NAME** manually.

b. Set the MODE to **STA** (Note: The Mode should always be STA as of the time of this implementation).

c. Tap the **ONBOARD** button to send the onboarding request to the MSO portal:



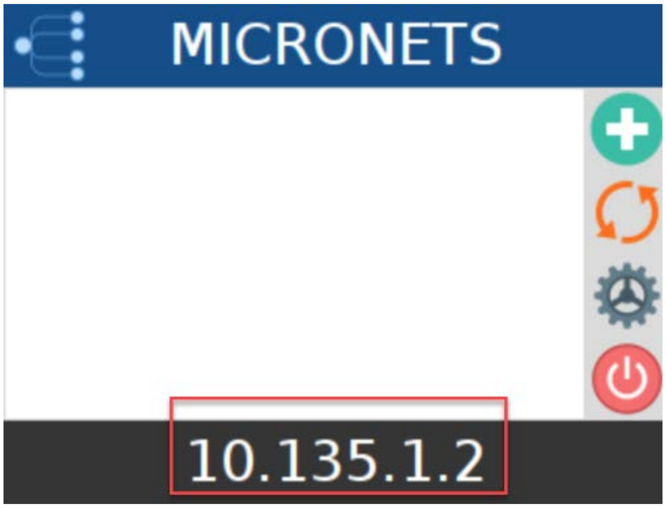
MAC: 00:C0:CA:97:D1:1F

MODE: ● STA ○ AP

CLASS: Generic

NAME: Pi1-nm1

8. On the Pi you will see the device has been onboarded to the Micronets Gateway and has received an IP address:



4.2.7 Interacting with Micronets Manager

The Micronets Manager, which is hosted in the cloud, has API endpoints exposed in order to allow implementers to manage the Micronets Gateway through the Micronets Manager service. This section describes how to set up postman and execute different functions.

4.2.7.1 Prerequisites

In order to successfully complete this section of the documentation, be sure to have completed the product installation section above and downloaded the postman application onto a laptop that has internet access: https://www.postman.com/downloads/.

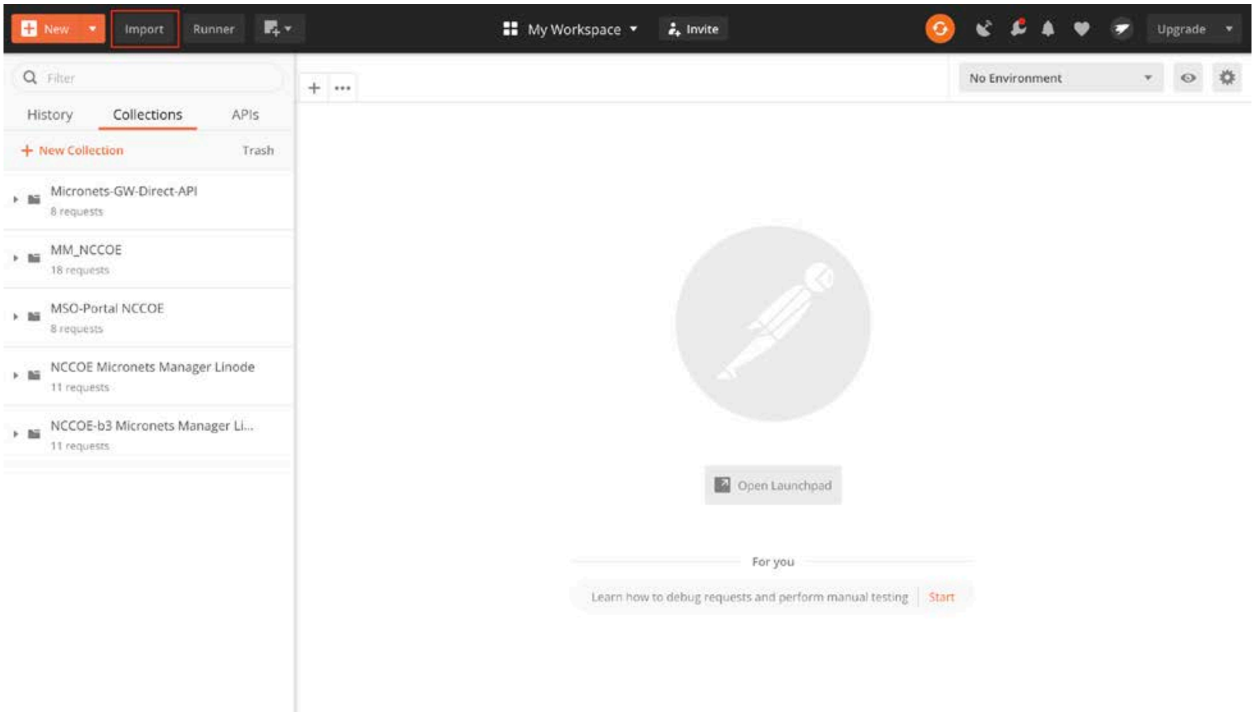
4.2.7.2 Instructions

1. Once Postman is installed and set up on the laptop, proceed to the following site to download the Micronets Manager Linode postman collections:

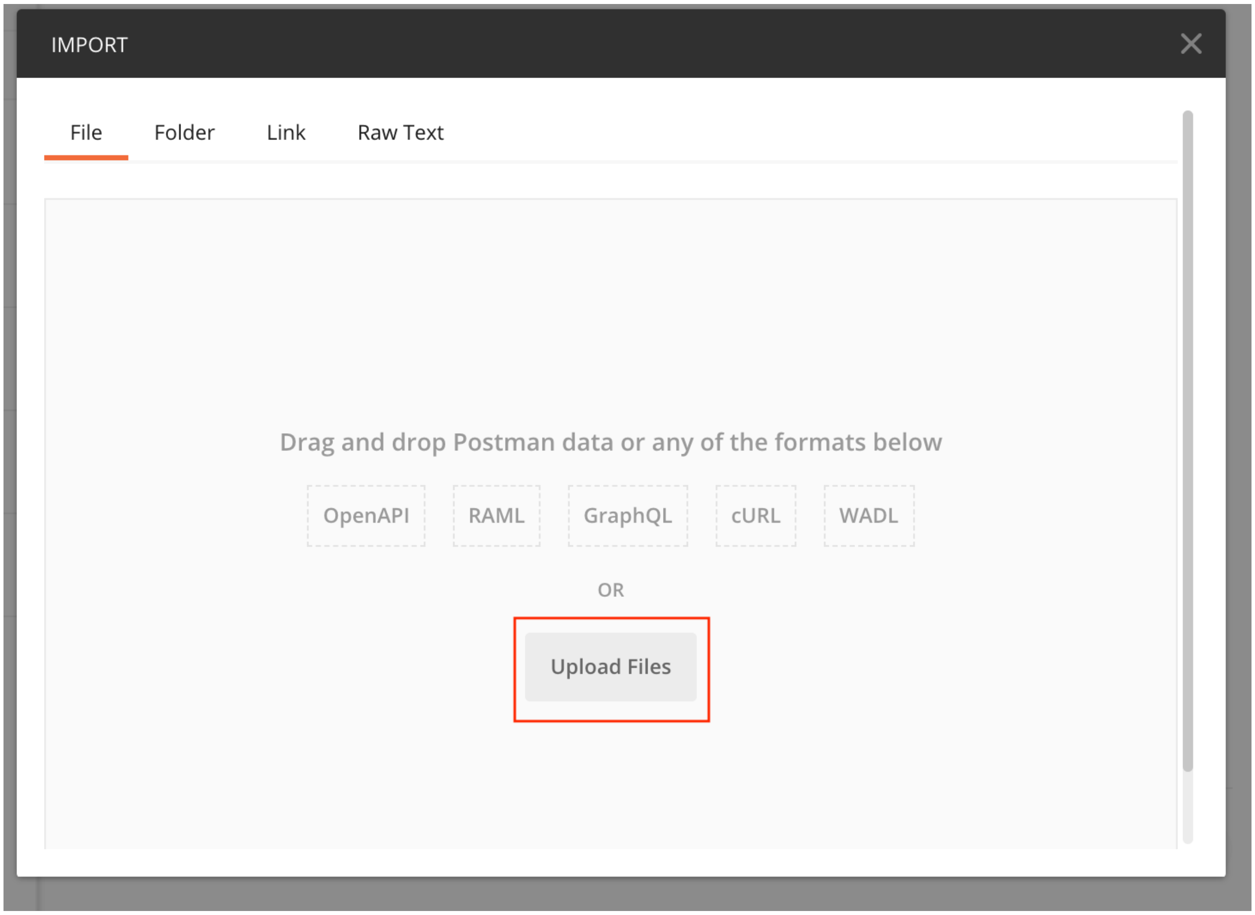
   Follow the links: https://raw.githubusercontent.com/cablelabs/micronets-manager/nccoe-build-3/scripts/Micronets_Manager_API.postman_collection.json

   https://raw.githubusercontent.com/cablelabs/micronets-manager/nccoe-build-3/scripts/Micronets_Manager_API.postman_globals.json

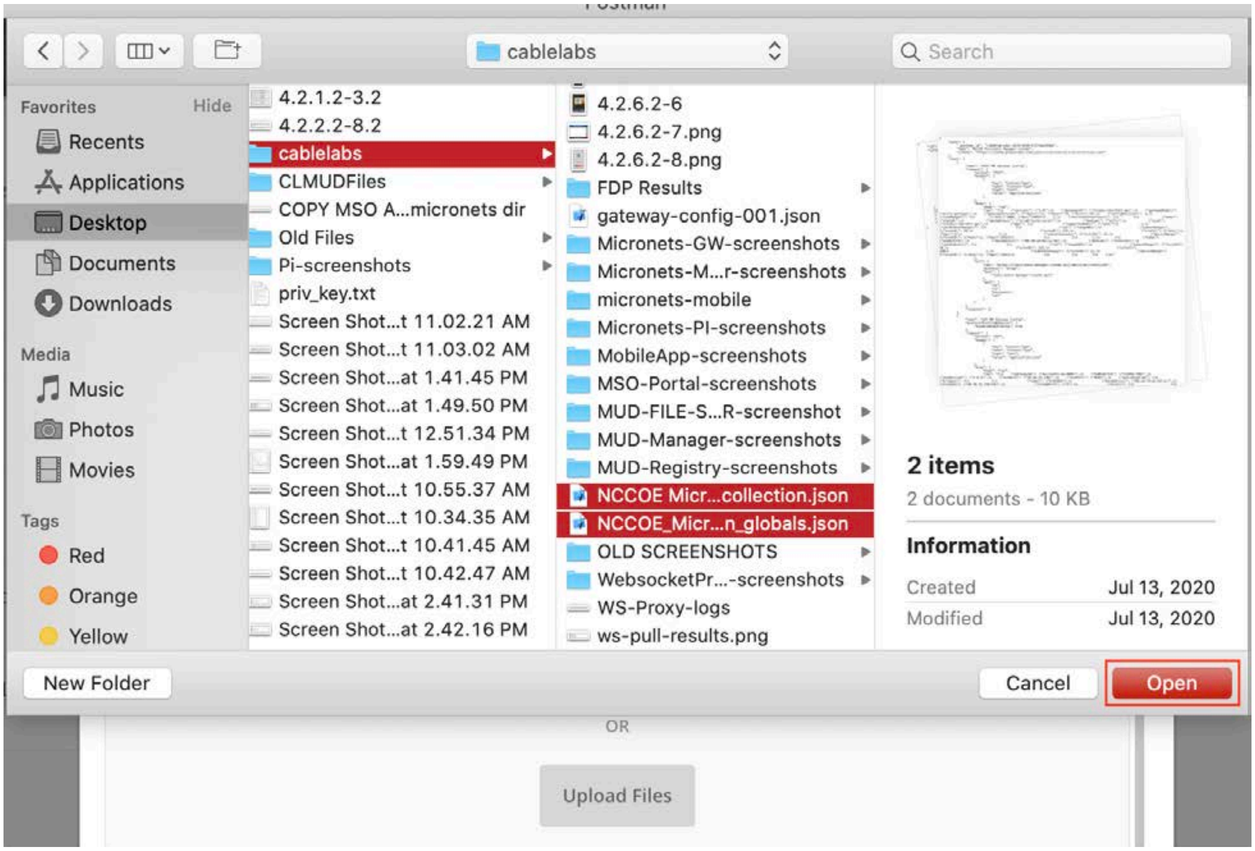
2. Open the postman application and sign in.
3. Click the import button to import the collections downloaded in step 1:



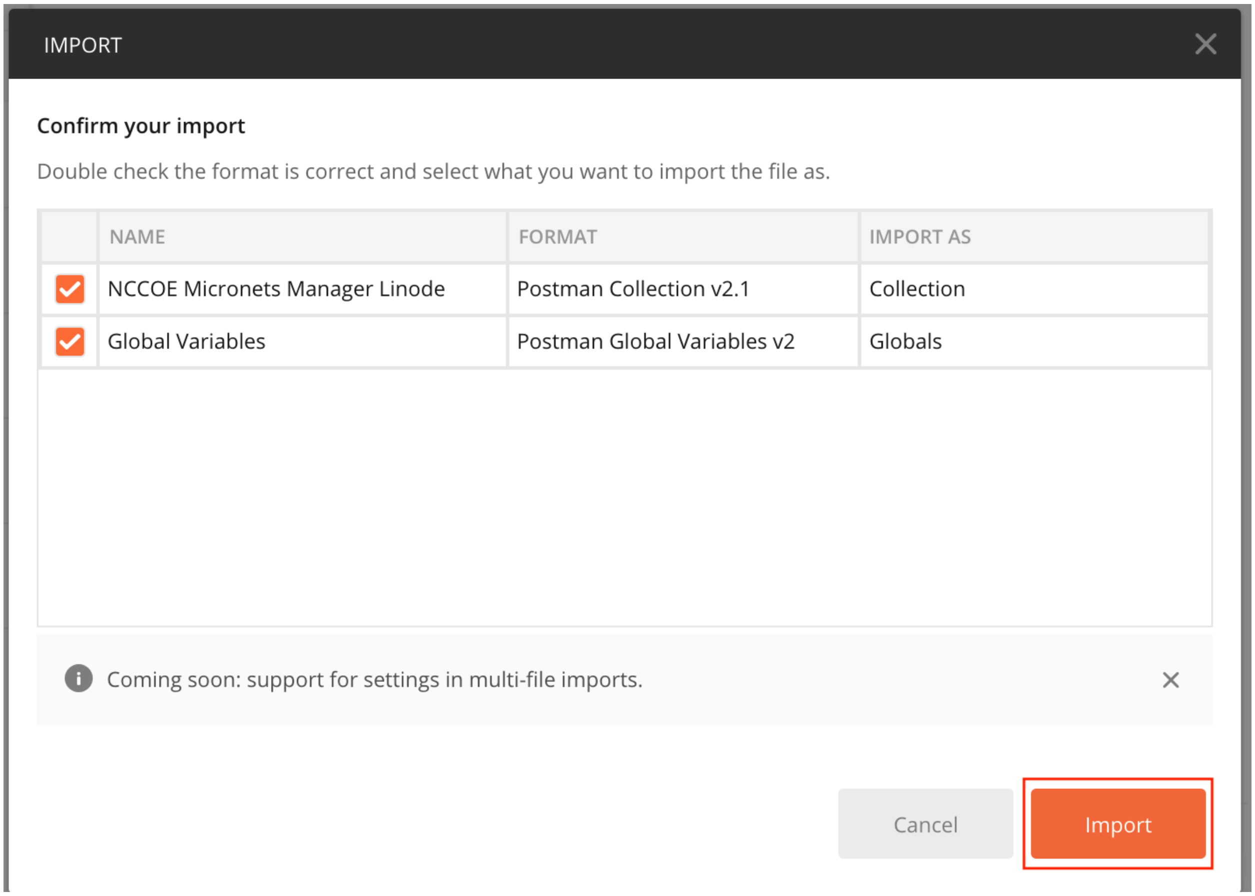
4. Next, click **upload files:**



5. Select the postman and global environmental variables collections downloaded in step 1:



6. Confirm your import and click **Import**:



IMPORT

Confirm your import

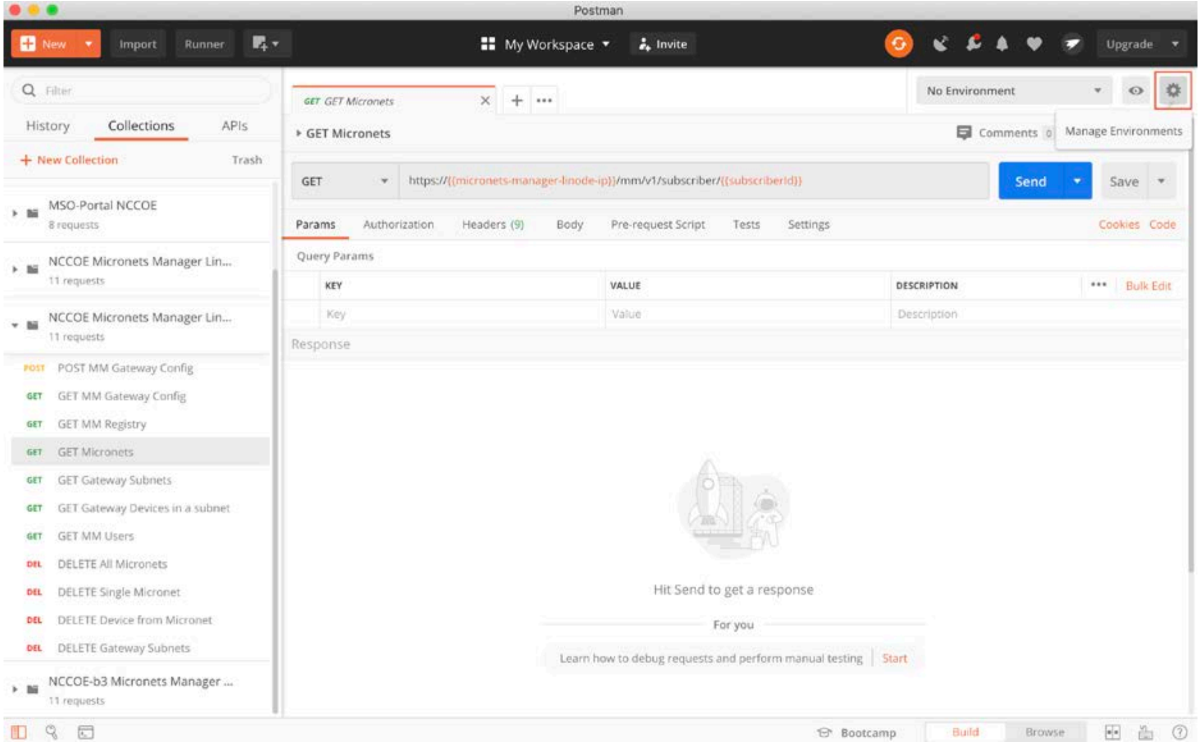
Double check the format is correct and select what you want to import the file as.

	NAME	FORMAT	IMPORT AS
☑	NCCOE Micronets Manager Linode	Postman Collection v2.1	Collection
☑	Global Variables	Postman Global Variables v2	Globals

Coming soon: support for settings in multi-file imports.

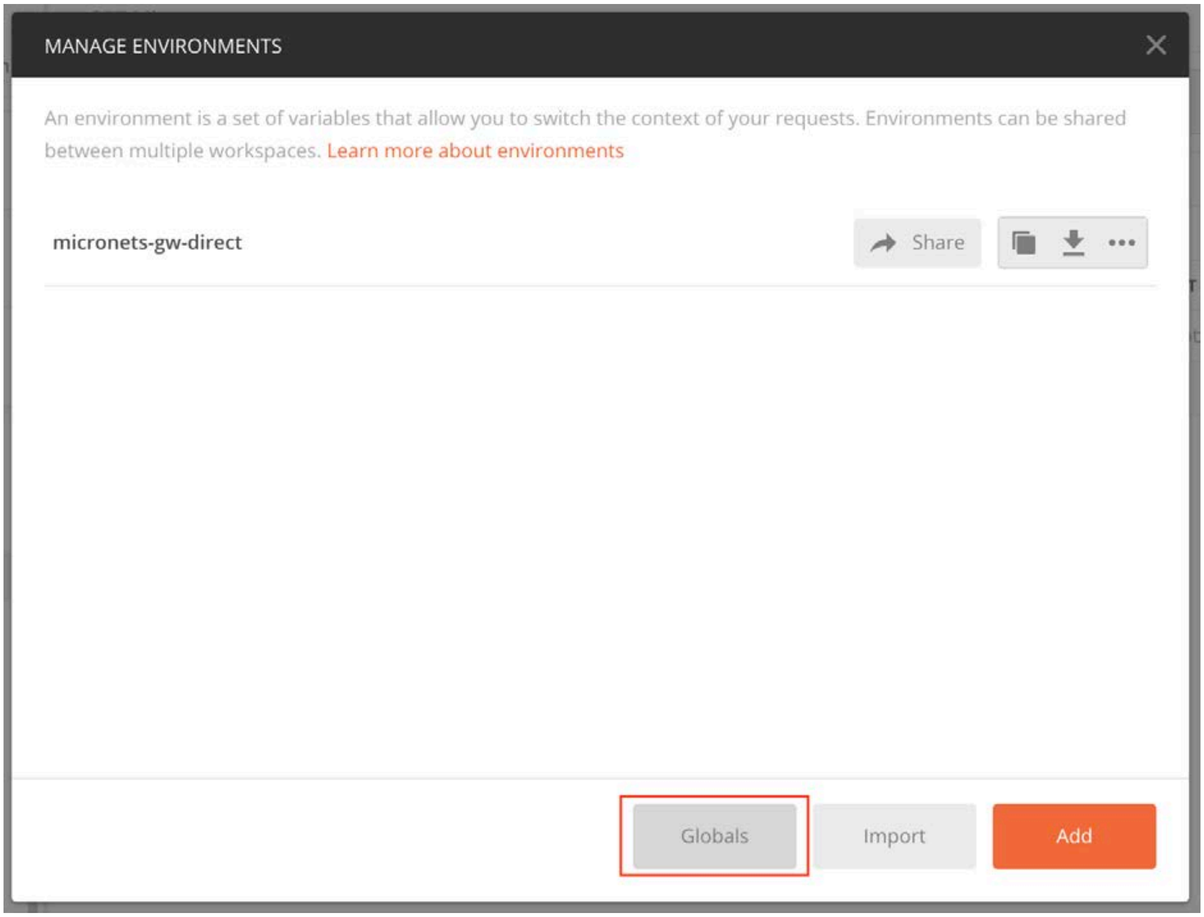
7. You will need to set the Globals for the micronets-manager-linode-ip, subscriberId and mso-portal-linode-ip:

   a. Click the gear button in the top right-hand corner of application to **Manage Environments**:



GET https://{{micronets-manager-linode-ip}}/mm/v1/subscriber/{{subscriberId}}

b. Click **Globals**:



MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

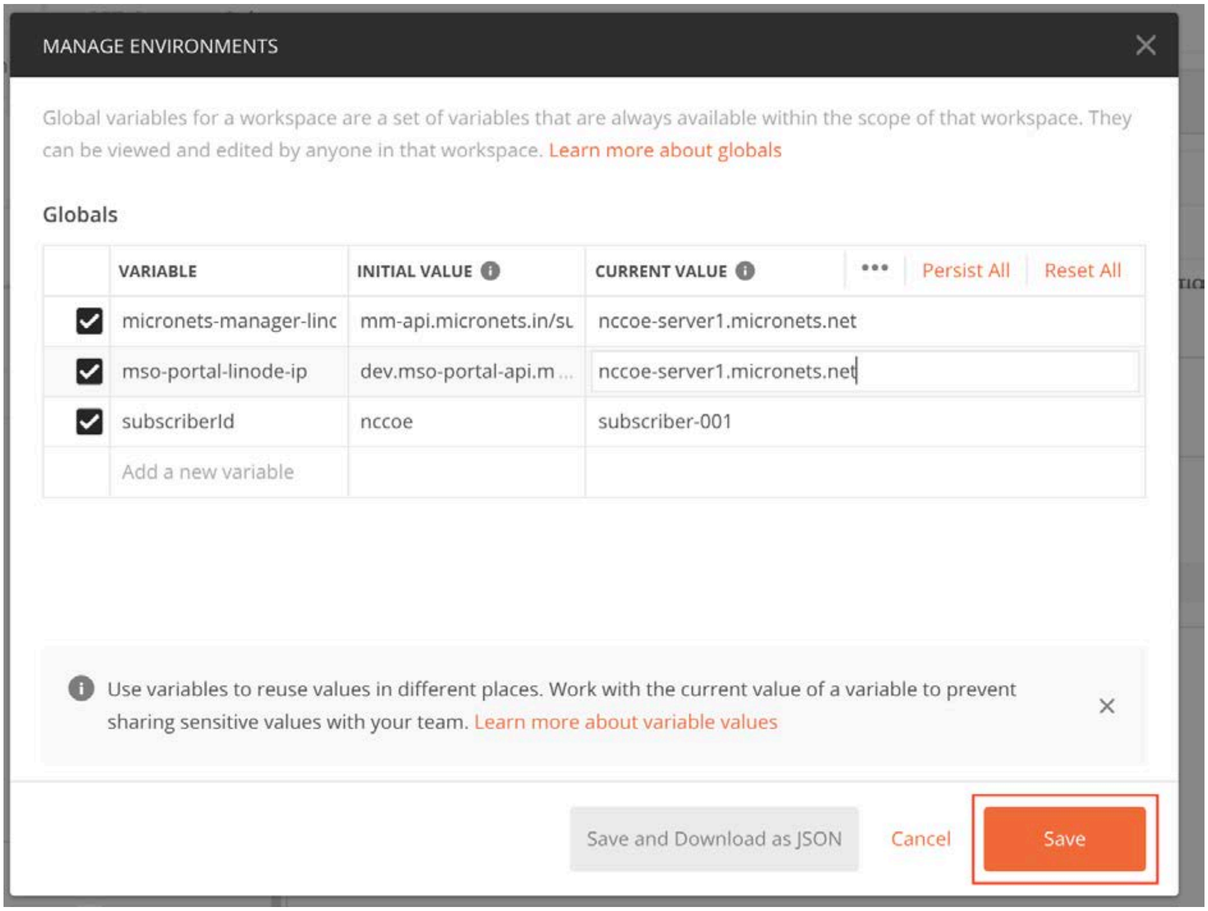
micronets-gw-direct

c. Modify the current values for the **micronets-manager-linode-ip, subscriberId and mso-portal-linode-ip** variables as follows and click **Save**:

micronets-manager-linode-ip: nccoe-server1.micronets.net

subscriberId: subscriber-001

mso-portal-linode-ip: nccoe-server1.micronets.net



MANAGE ENVIRONMENTS

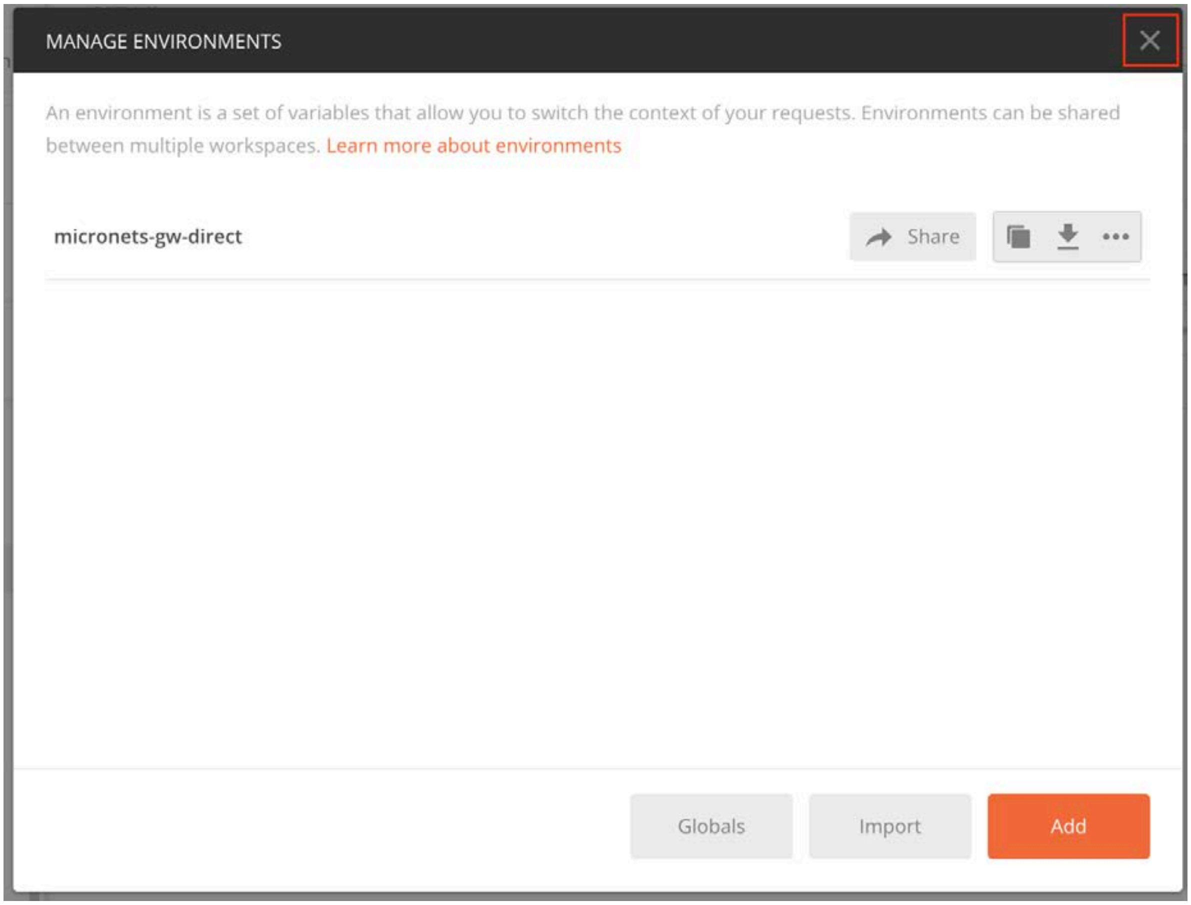
Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals

Globals

	VARIABLE	INITIAL VALUE	CURRENT VALUE
☑	micronets-manager-linc	mm-api.micronets.in/su	nccoe-server1.micronets.net
☑	mso-portal-linode-ip	dev.mso-portal-api.m ...	nccoe-server1.micronets.net
☑	subscriberId	nccoe	subscriber-001
	Add a new variable		

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

d. Exit out of the menu:



MANAGE ENVIRONMENTS

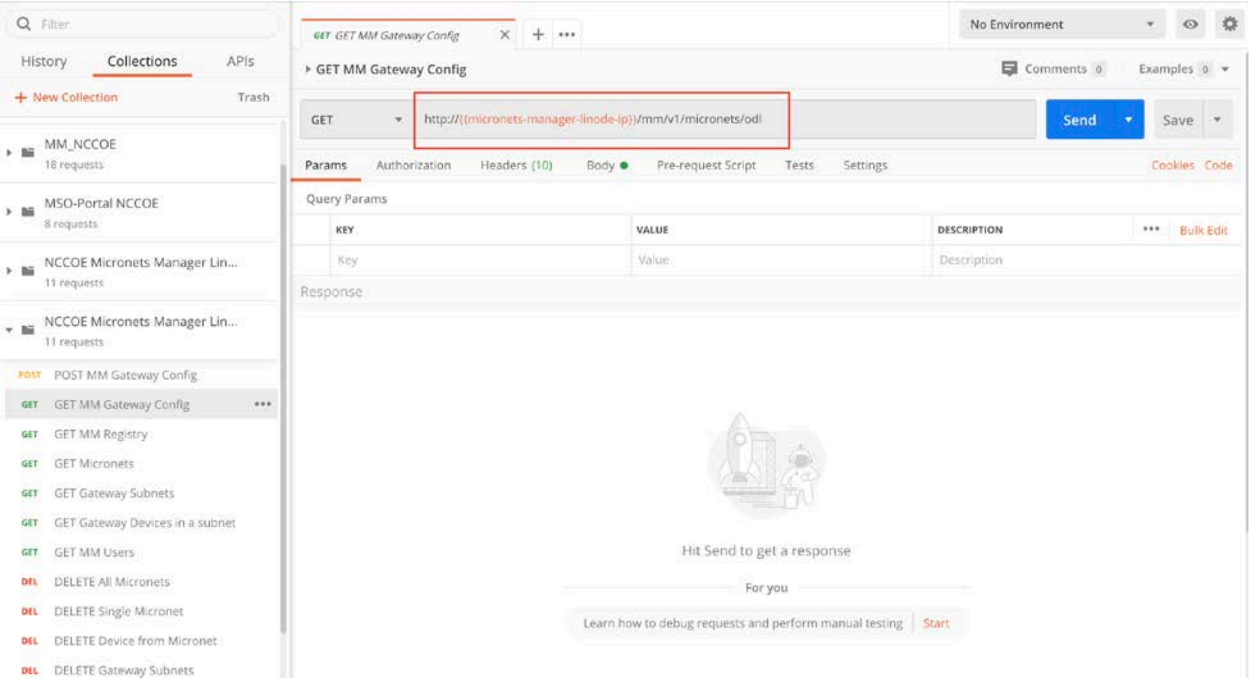
An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

micronets-gw-direct

Next, open the postman collection and review and modify the URLs for the calls to ensure the API endpoint paths match your implementation:

a. Modify the **GET MM Gateway Config** command to reflect the following. Executing this command will pull the current Gateway config from the Micronets Manager:

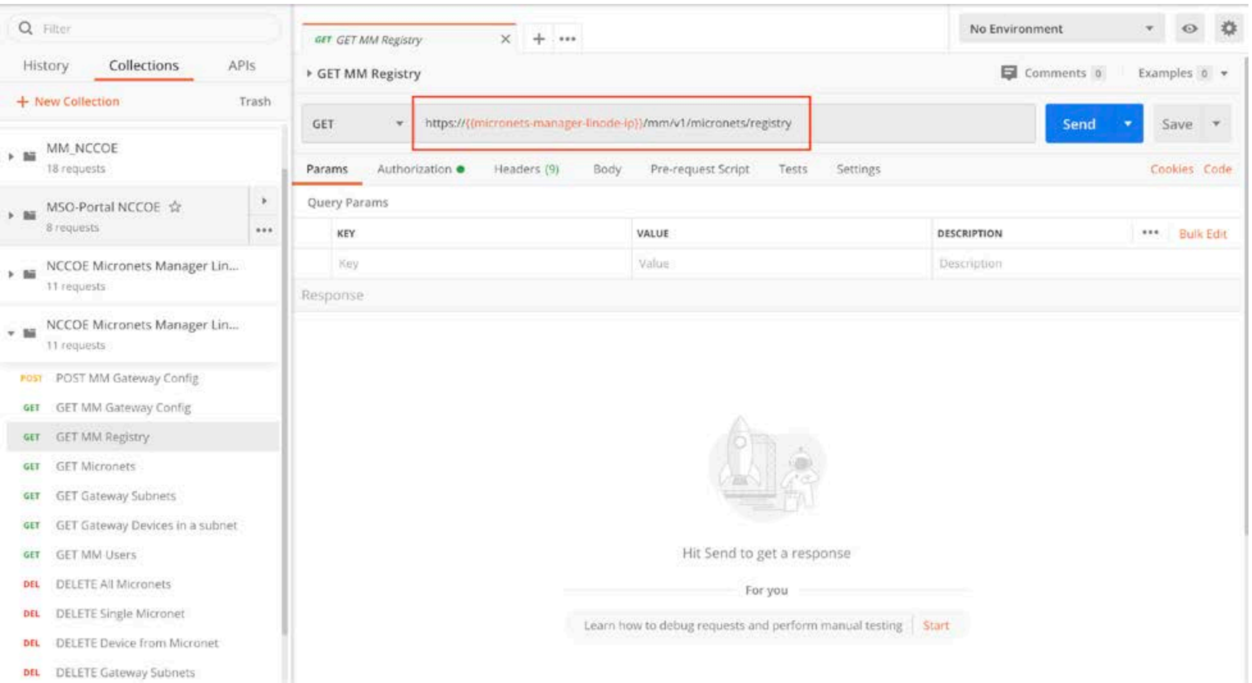
```
http://{{micronets-manager-linode-ip}}/mm/v1/micronets/odl
```



```
http://{{micronets-manager-linode-ip}}/mm/v1/micronets/odl
```

b. Modify the **GET MM Registry** command to reflect the following. Executing this command will pull the current registry from the Micronets Manager:

```
https://{{micronets-manager-linode-ip}}/mm/v1/micronets/registry
```

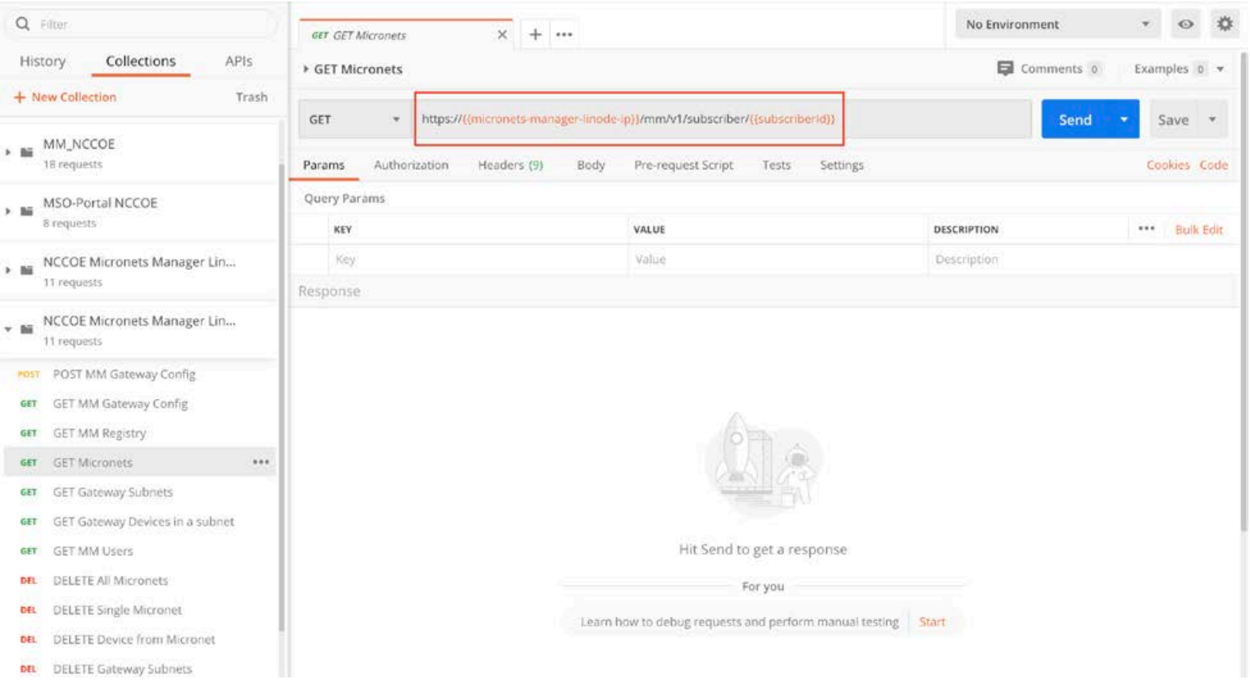


GET https://{{micronets-manager-linode-ip}}/mm/v1/micronets/registry

KEY	VALUE	DESCRIPTION
Key	Value	Description

Modify the **GET Micronets** command to reflect the following. Executing this command will pull a list of the current micronets on the Gateway from the Micronets Manager:

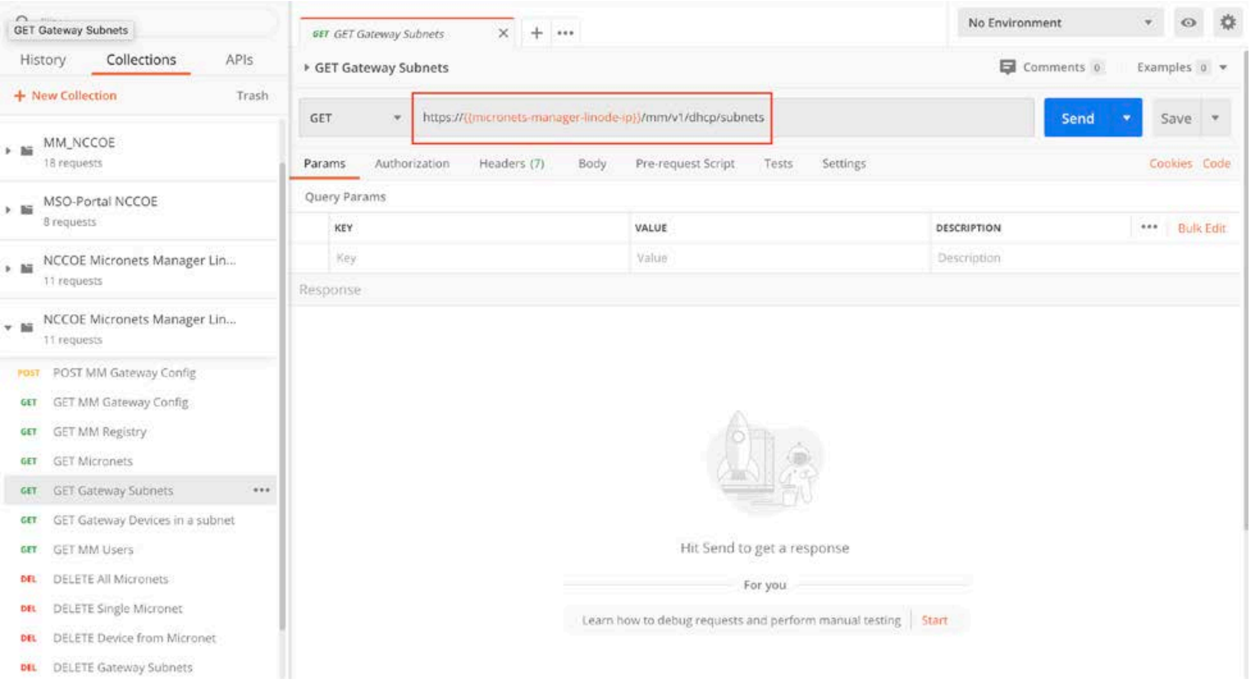
```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/subscriber/{{subscriberId}}
```



GET https://{{micronets-manager-linode-ip}}/mm/v1/subscriber/{{subscriberId}}

d. Modify the **GET Gateway Subnets** command to reflect the following. Executing this command will pull a list of the current subnets on the Gateway from the Micronets Manager:

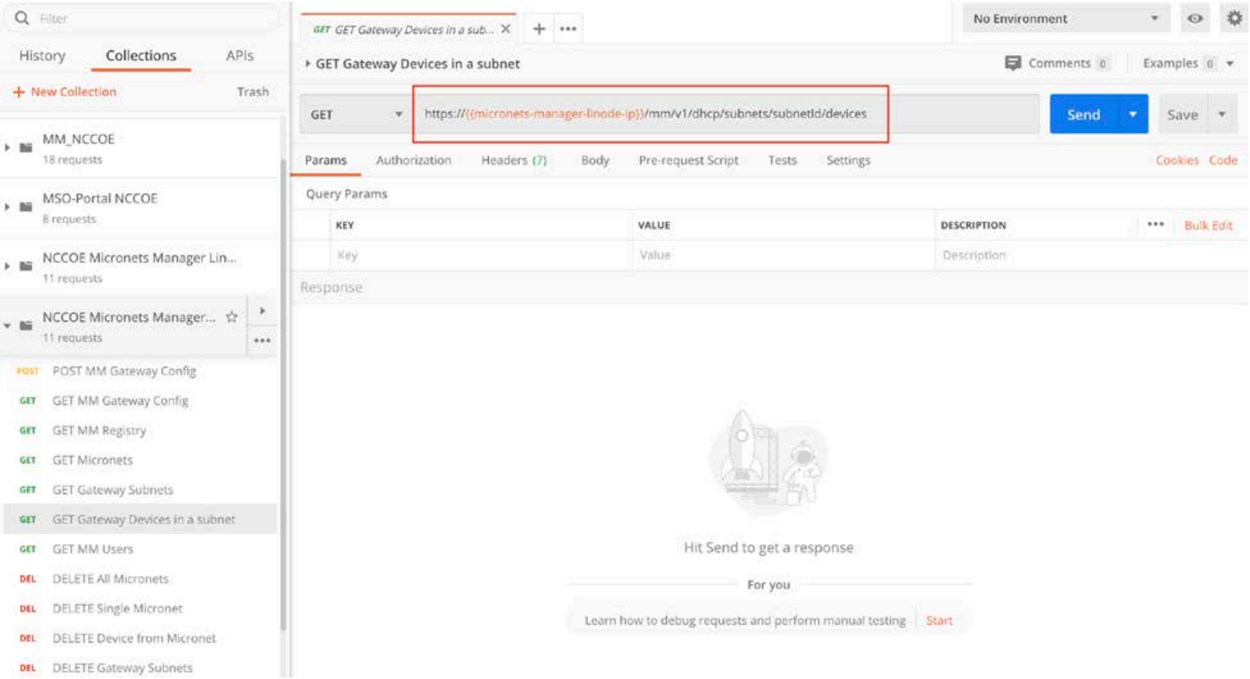
```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/dhcp/subnets
```



GET https://{{micronets-manager-linode-ip}}/mm/v1/dhcp/subnets

e. Modify the **GET Gateway Devices in a subnet** command to reflect the following. Executing this command will pull a list of the current devices in a subnet on the Gateway from the Micronets Manager:

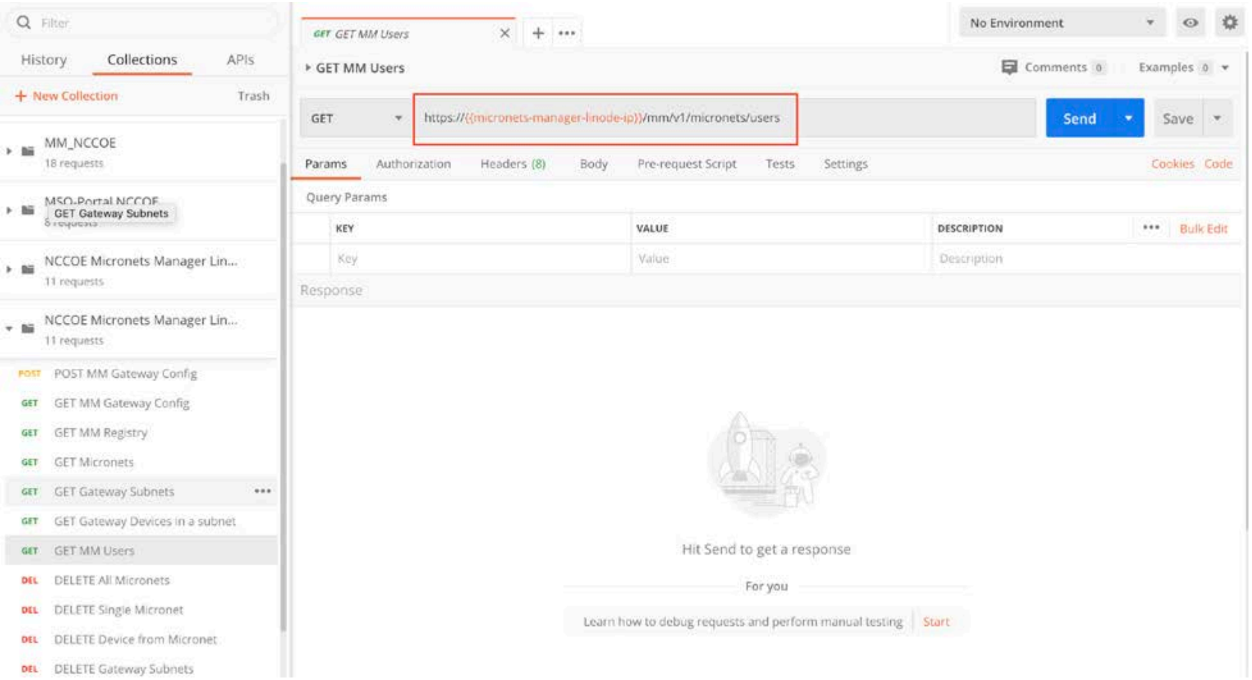
```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/dhcp/subnets/subnetId/devices
```



GET https://{{micronets-manager-linode-ip}}/mm/v1/dhcp/subnets/subnetId/devices

f. Modify the **GET MM Users** command to reflect the following. Executing this command will pull a list of the users associated with the subscriber ID from the Micronets Manager:

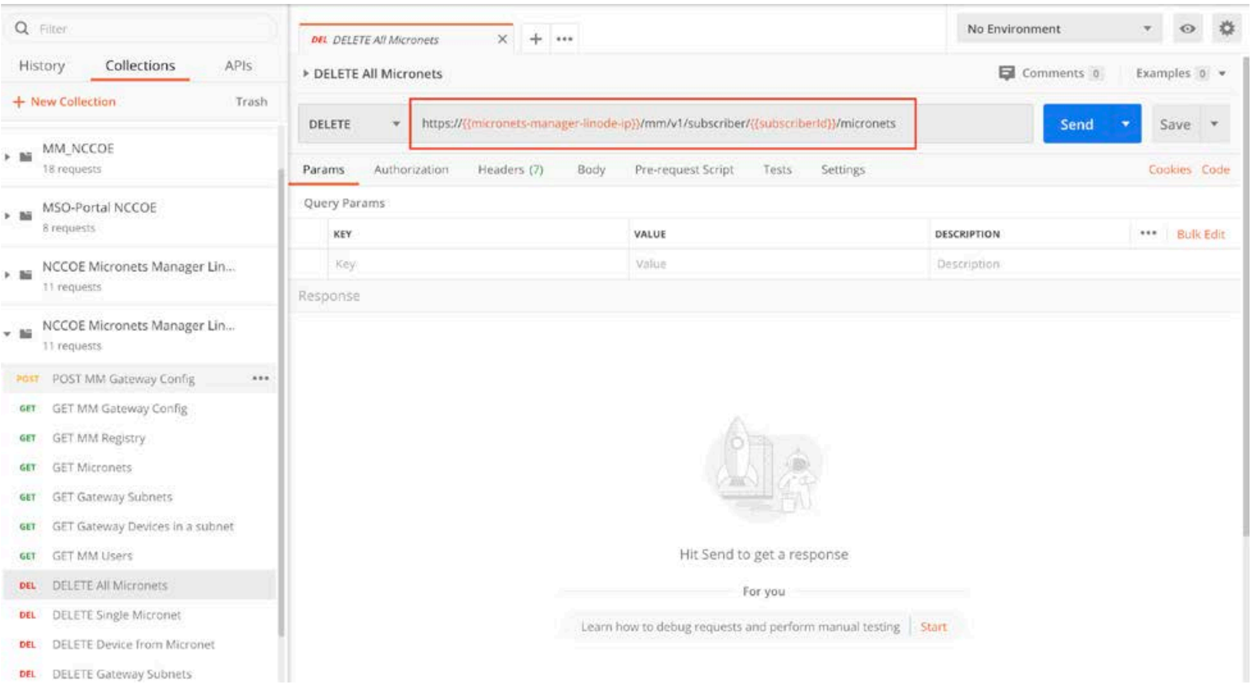
```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/micronets/users
```



GET https://{{micronets-manager-linode-ip}}/mm/v1/micronets/users

g. Modify the **DELETE All Micronets** command to reflect the following. Executing this command will delete all of the current micronets on the Gateway via the Micronets Manager:

```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/subscriber/{{subscriberId}}/micronets
```



DELETE All Micronets

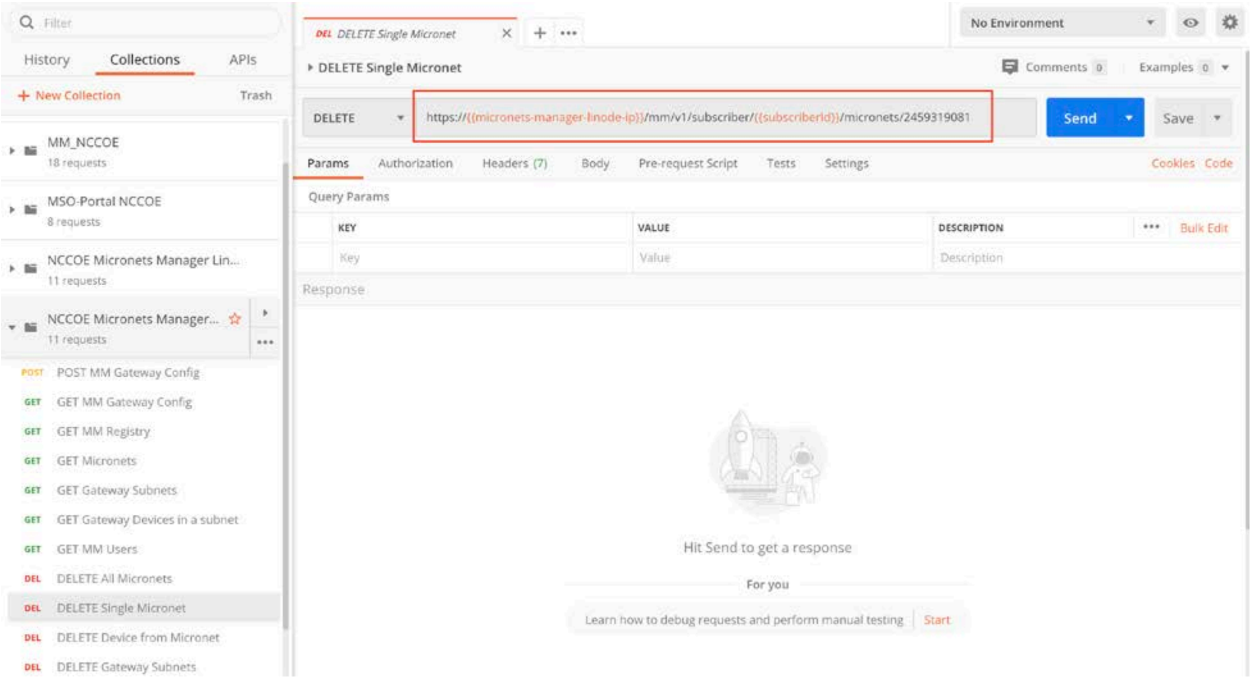
DELETE https://{{micronets-manager-linode-ip}}/mm/v1/subscriber/{{subscriberId}}/micronets

h. Modify the **DELETE Single Micronets** command to reflect the following. Executing this command will delete a specific micronet on the Gateway via the Micronets Manager. This command is to be modified before executing to specify the **<micronetID>**, which can be retrieved by executing the GET Micronets command:

https://{{micronets-manager-linode-ip}}/sub/{{subscriberId}}/api/mm/v1/subscriber/{{subscriberId}}/micronets/**<micronetID>**

Below is an example of this command:

```
https://{{micronets-manager-linode-ip}}/sub/{{subscriberId}}/api/mm/v1/subscriber/{{subscriberId}}/micronets/2453819029
```



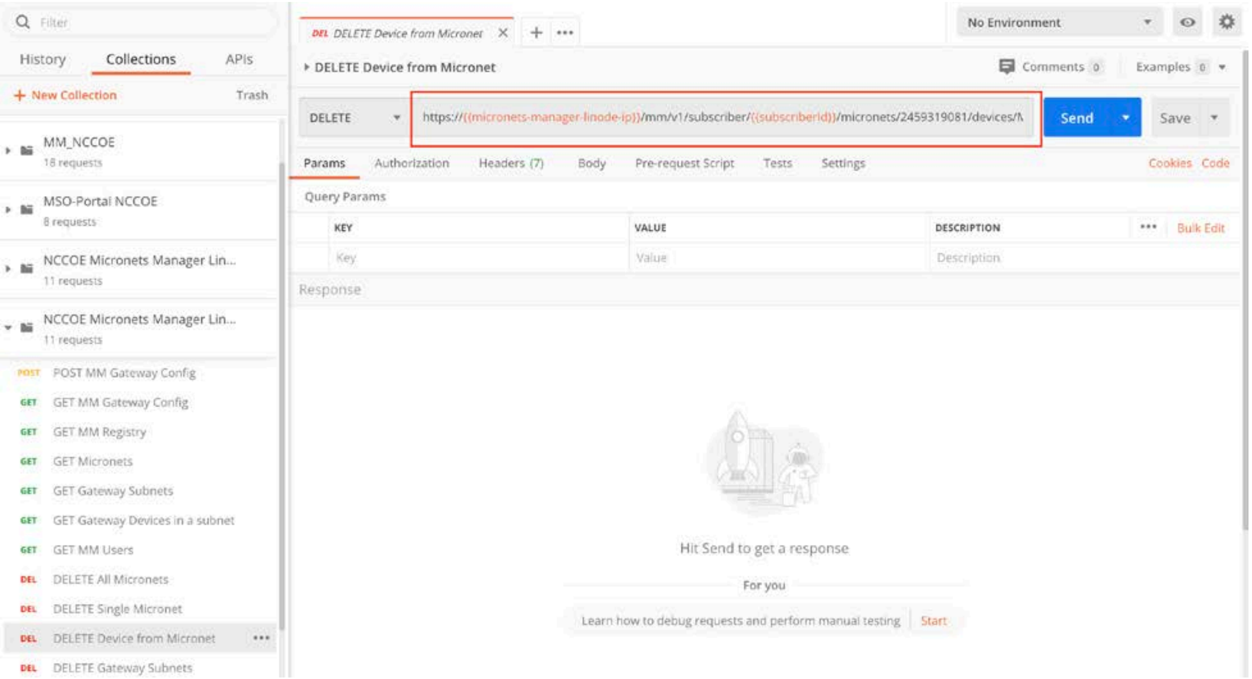
DELETE https://{{micronets-manager-linode-ip}}/mm/v1/subscriber/{{subscriberId}}/micronets/2459319081

i. Modify the **DELETE Device from Micronet** command to reflect the following. Executing this command will delete a specific device from a particular micronet on the Gateway via the Micronets Manager. This command is to be modified before executing to specify the **<micronetID>** and **<deviceID>**, which can be retrieved by executing the GET Micronets command:

https://{{micronets-manager-linode-ip}}/sub/{{subscriberId}}/api/mm/v1/subscriber/{{subscriberId}}/micronets/**<micronetID>** /devices/**<deviceID>**

Below is an example of this command:

```
https://{{micronets-manager-linode-ip}}/sub/{{subscriberId}}/api/mm/v1/subscriber/{{subscriberId}}/micronets/2136369149/devices/da34c7219c2c97f0e2c2838e66c725d137f3c097
```

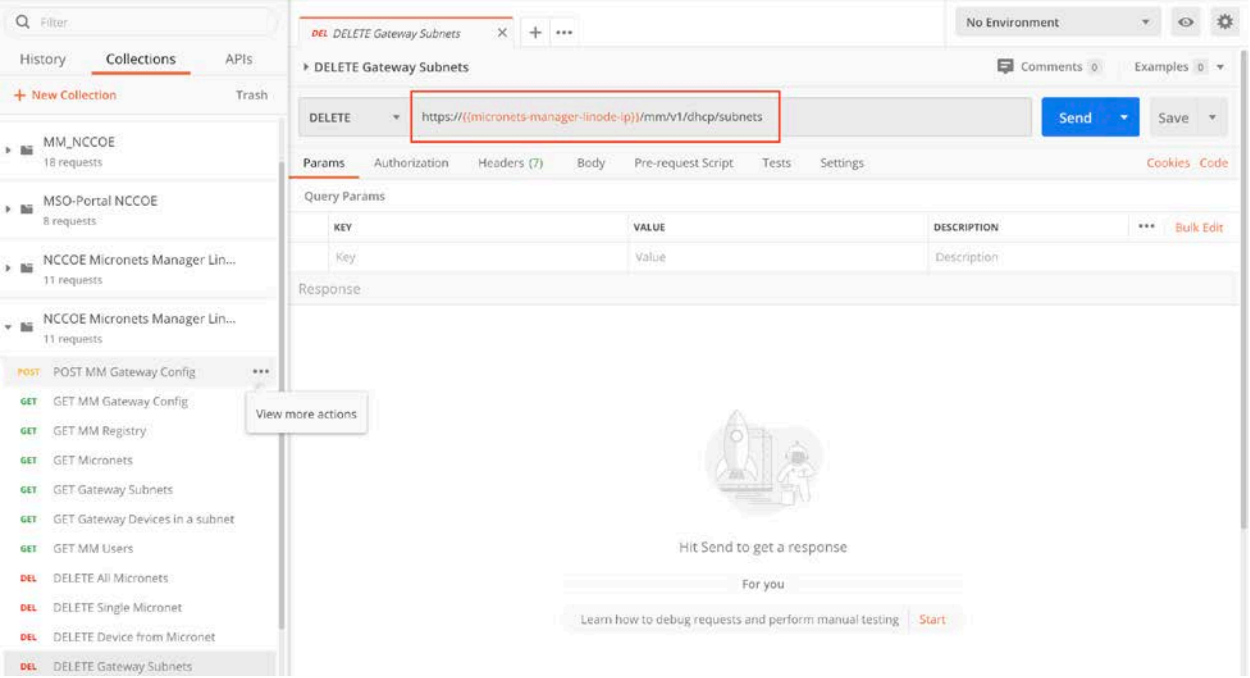


DELETE Device from Micronet

DELETE https://{{micronets-manager-linode-ip}}/mm/v1/subscriber/{{subscriberId}}/micronets/2459319081/devices/N

j. Modify the **DELETE Gateway Subnets** command to reflect the following. Executing this command will delete all subnets on the Gateway via the Micronets Manager:

```
https://{{micronets-manager-linode-ip}}/sub/{{subscrib-
erId}}/api/mm/v1/dhcp/subnets
```



DELETE Gateway Subnets

DELETE https://{{micronets-manager-linode-ip}}/mm/v1/dhcp/subnets

4.2.8 Removing Micronets Proto-Pi from a Micronet

Removing a Micronets Proto-Pi from a micronet will remove the network credentials from the device. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation: https://github.com/cablelabs/micronets/blob/nccoe-build-3/docs/operation/pi-offboarding.md.

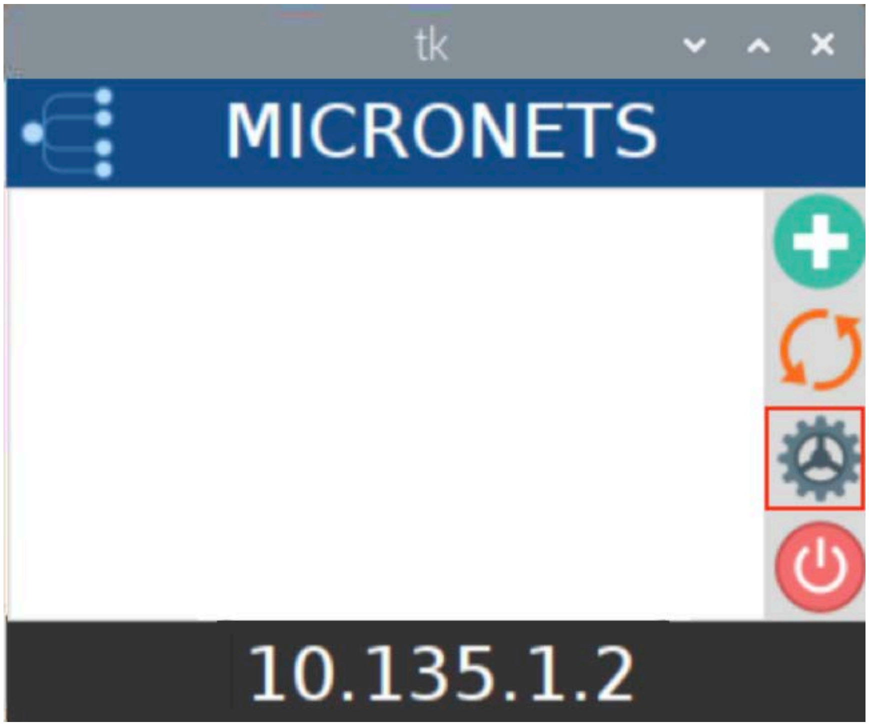
4.2.8.1 Prerequisites

To successfully complete this section, the following are required:

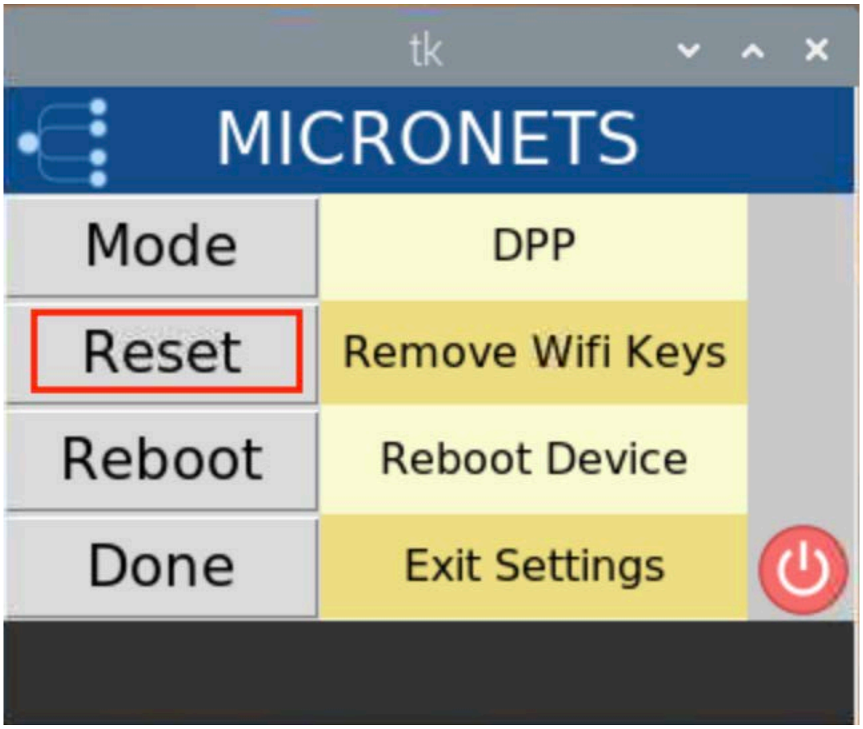
- a Raspberry Pi with the Micronets Proto-Pi software installed and configured
- a device that is currently onboarded to the Micronets Gateway

4.2.8.2 Instructions:

1. Power on the Micronets Proto-Pi device.
2. Tap Settings:

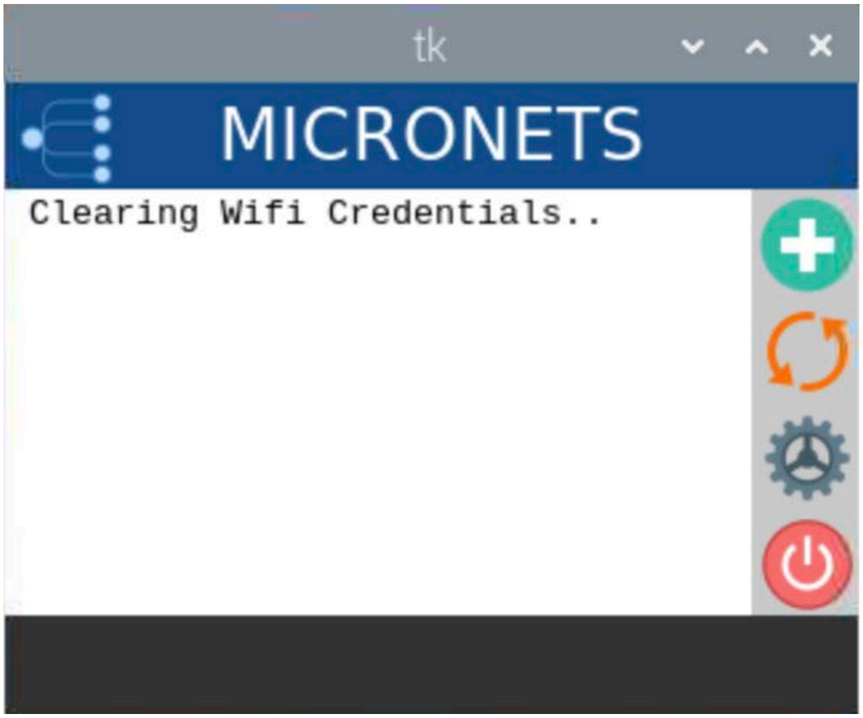


3. Tap Reset:



3322

3323 You should see output similar to the following:



```
Clearing Wifi Credentials..
```

3324

4.2.9 Removing an MSO Subscriber

Removing a subscriber involves removing the subscriber from the MSO portal database, removing the subscriber's micronets, and removing the subscriber's Micronets Manager. For additional instructions not detailed in this documentation, please follow the link to the CableLabs documentation: https://github.com/cablelabs/micronets/blob/nccoe-build-3/docs/operation/pi-offboarding.md.

4.2.9.1 Prerequisites

To successfully complete this section be sure to have completed both the product installation section and . Ensure all steps have been successfully completed before proceeding to the instructions.

4.2.9.2 Instructions

1. Remove the subscriber from the MSO portal using:

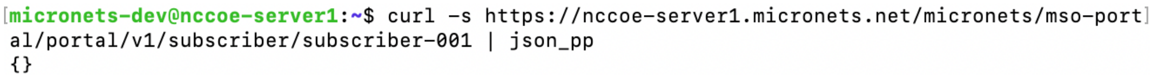
   ```
   curl -s -X DELETE https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/subscriber/subscriber-001 | json_pp
   ```

2. Verify that the subscriber is removed from the MSO portal by executing the following commands:

   a. Check if the subscriber ID is present in the subscriber list:

      ```
      curl -s https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/subscriber/subscriber-001
      ```

      You should see output similar to the following:

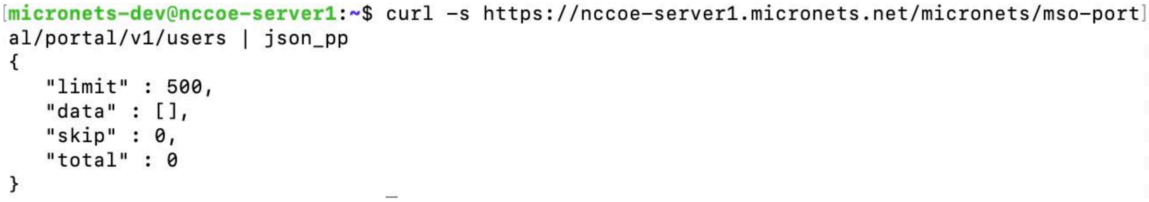
      

      ```
      micronets-dev@nccoe-server1:~$ curl -s https://nccoe-server1.micronets.net/micronets/mso-port
      al/portal/v1/subscriber/subscriber-001 | json_pp
      {}
      ```

   b. Next, check if the user is present in the list of users in the MSO portal:

      ```
      curl -s https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/users | json_pp
      ```

      You should see output similar to the following:

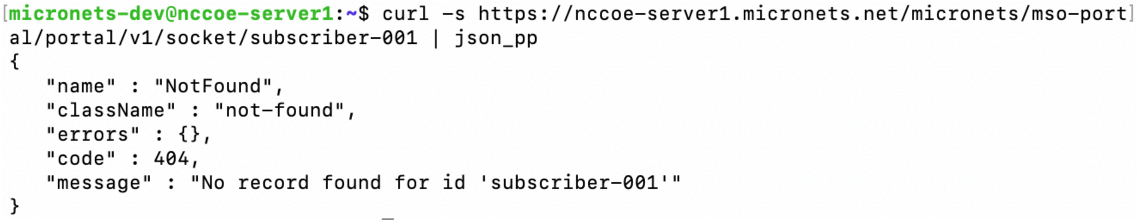


```
micronets-dev@nccoe-server1:~$ curl -s https://nccoe-server1.micronets.net/micronets/mso-port
al/portal/v1/users | json_pp
{
   "limit" : 500,
   "data" : [],
   "skip" : 0,
   "total" : 0
}
```

c. Finally, check to see if there is a socket present for the subscriber ID:

```
curl -s https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/socket/subscriber-001 | json_pp
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -s https://nccoe-server1.micronets.net/micronets/mso-port
al/portal/v1/socket/subscriber-001 | json_pp
{
   "name" : "NotFound",
   "className" : "not-found",
   "errors" : {},
   "code" : 404,
   "message" : "No record found for id 'subscriber-001'"
}
```

Note: There could be scenarios where the commands above do not show empty lists. If that is the case, the subscriber has not been deleted properly. You can delete the subscriber entries in the MSO portal subtables by executing the following commands:

d. Delete the subscriber ID from the user list manually:

```
curl -s -X DELETE https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/users/subscriber-001 | json_pp
```

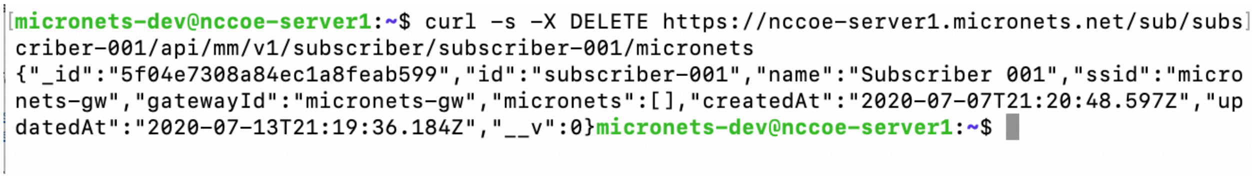
e. Delete the subscriber ID from the socket list manually:

```
curl -s -X DELETE https://nccoe-server1.micronets.net/micronets/mso-portal/portal/v1/socket/subscriber-001
```

3. Remove all the micronets for the subscriber using:

```
curl -s -X DELETE https://nccoe-server1.micronets.net/sub/subscriber-001/api/mm/v1/subscriber/subscriber-001/micronets
```

You should see output similar to the following:



```
micronets-dev@nccoe-server1:~$ curl -s -X DELETE https://nccoe-server1.micronets.net/sub/subscriber-001/api/mm/v1/subscriber/subscriber-001/micronets
{"_id":"5f04e7308a84ec1a8feab599","id":"subscriber-001","name":"Subscriber 001","ssid":"micronets-gw","gatewayId":"micronets-gw","micronets":[],"createdAt":"2020-07-07T21:20:48.597Z","updatedAt":"2020-07-13T21:19:36.184Z","__v":0}micronets-dev@nccoe-server1:~$ 
```

This will remove the micronets on the connected Micronets Gateway. If the gateway is not connected to its peer Micronets Manager, the micronets can be deleted directly on the gateway using:

```
curl -s -X DELETE http://localhost:5000/micronets/v1/gateway/micronets
```

4. You can verify that the micronets have been deleted by running:

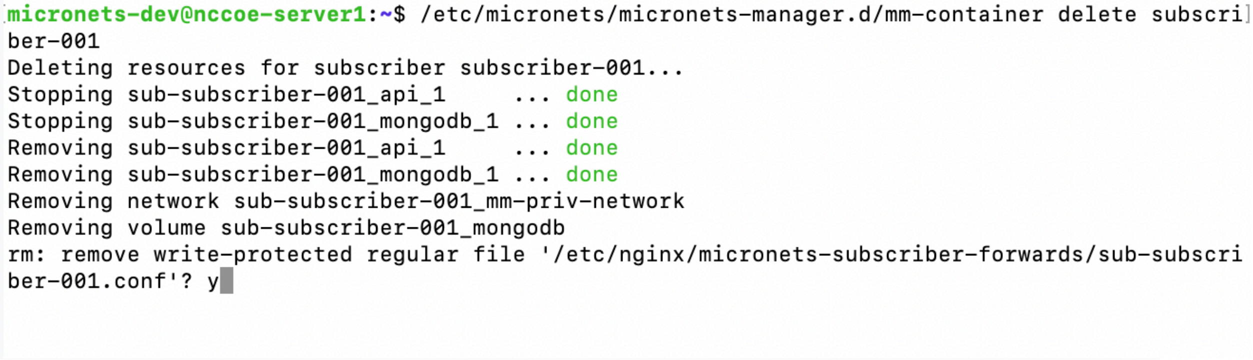
```
curl -s https://nccoe-server1.micronets.net/sub/subscriber-001/api/mm/v1/subscriber/subscriber-001/micronets
```

This should return an empty micronets list.

5. Remove the Micronets Manager docker container for a subscriber by running:

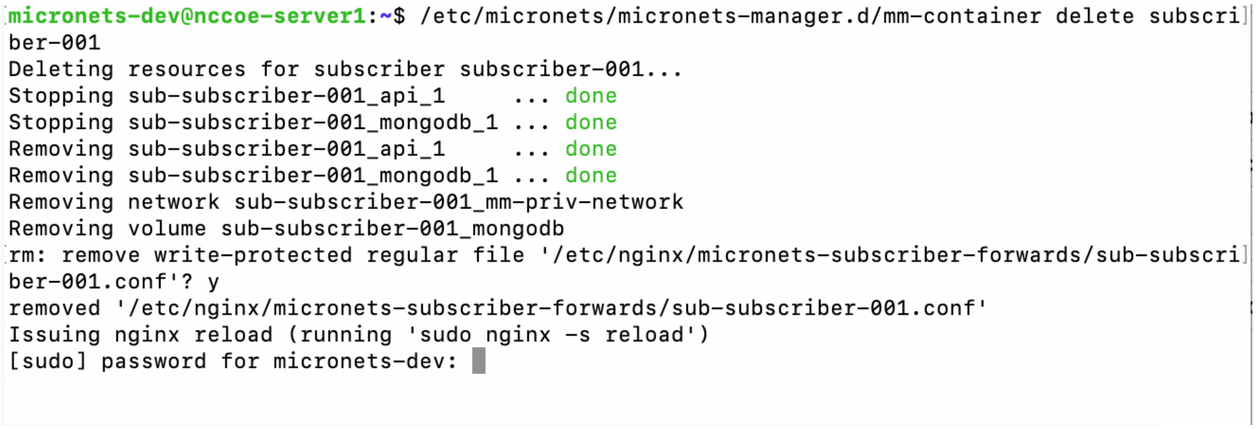
```
/etc/micronets/micronets-manager.d/mm-container delete subscriber-001
```

You will prompted to remove the config file:



```
micronets-dev@nccoe-server1:~$ /etc/micronets/micronets-manager.d/mm-container delete subscriber-001
Deleting resources for subscriber subscriber-001...
Stopping sub-subscriber-001_api_1     ... done
Stopping sub-subscriber-001_mongodb_1 ... done
Removing sub-subscriber-001_api_1     ... done
Removing sub-subscriber-001_mongodb_1 ... done
Removing network sub-subscriber-001_mm-priv-network
Removing volume sub-subscriber-001_mongodb
rm: remove write-protected regular file '/etc/nginx/micronets-subscriber-forwards/sub-subscriber-001.conf'? y
```

Lastly, you will be prompted to provide sudo privileges:



```
micronets-dev@nccoe-server1:~$ /etc/micronets/micronets-manager.d/mm-container delete subscri
ber-001
Deleting resources for subscriber subscriber-001...
Stopping sub-subscriber-001_api_1     ... done
Stopping sub-subscriber-001_mongodb_1 ... done
Removing sub-subscriber-001_api_1     ... done
Removing sub-subscriber-001_mongodb_1 ... done
Removing network sub-subscriber-001_mm-priv-network
Removing volume sub-subscriber-001_mongodb
rm: remove write-protected regular file '/etc/nginx/micronets-subscriber-forwards/sub-subscri
ber-001.conf'? y
removed '/etc/nginx/micronets-subscriber-forwards/sub-subscriber-001.conf'
Issuing nginx reload (running 'sudo nginx -s reload')
[sudo] password for micronets-dev:
```

6. Confirm the Micronets Manager for the subscriber is removed by executing the following command:

```
curl -s https://nccoe-server1.micronets.net/sub/subscriber-001/api/mm/v1/subscriber/subscriber-001
```

5 Build 4 Product Installation Guides

This section of the practice guide contains detailed instructions for installing and configuring the products used to implement Build 4. For additional details on Build 4’s logical and physical architectures, please refer to NIST SP 1800-15B.

5.1 NIST SDN Controller/MUD Manager

5.1.1 NIST SDN Controller/MUD Manager Overview

This is a limited implementation that is intended to introduce a MUD manager build on top of an SDN controller. Build 4 implements all the abstractions in the MUD specification. At testing, this build uses strictly IPv4, and DHCP is the only standardized mechanism that it supports to associate MUD URLs with devices.

Build 4 uses a MUD manager built on the OpenDaylight SDN controller. This build works with IoT devices that emit their MUD URLs through DHCP. The MUD manager works by snooping the traffic passing through the controller to detect the emission of a MUD URL. The MUD URL extracted by the MUD manager is then used to retrieve the MUD file and corresponding signature file associated with the MUD URL. The signature file is used to verify the legitimacy of the MUD file. The MUD manager then translates the access control entries in the MUD file into flow rules that are pushed to the switch.

5.1.2 Configuration Overview

The following subsections document the software, hardware, and network configurations for the Build 4 SDN controller/MUD manager.

5.1.2.1 Hardware Configuration

This build requires installing the SDN controller/MUD manager on a server with at least two gigabytes of random access memory. This server must connect to at least one SDN-capable switch or router on the network, which is the MUD policy enforcement point. The MUD manager works with any OpenFlow 1.3-enabled SDN switch. For this implementation, a Northbound Networks Zodiac WX wireless SDN access point was used as the SDN switch.

5.1.2.2 Network Configuration

The SDN controller/MUD manager instance was installed and configured on a dedicated machine leveraged for hosting virtual machines in the Build 4 lab environment. The SDN controller/MUD manager listens on port 6653 for Open vSwitch (OVS) inbound connections, which are initiated by the OVS instance running on the Northbound Networks access point.

5.1.2.3 Software Configuration

For this build, the SDN controller/MUD manager was installed on an Ubuntu 18.04.01 64-bit server.

The SDN controller/MUD manager requires the following installations and components:

- Java SE Development Kit 8
- Apache Maven 3.5 or higher

5.1.3 Preinstallation

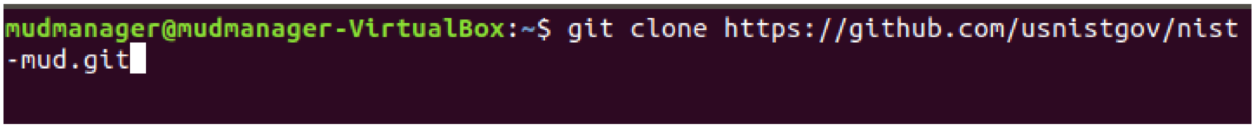
Build 4’s GitHub page provides documentation that was followed to complete this section: https://github.com/usnistgov/nist-mud.

- Install JDK 1.8: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.
- Install Maven 3.5 or higher: https://maven.apache.org/download.cgi.

5.1.4 Setup

1. Execute the following command to clone the Git project:

```
git clone https://github.com/usnistgov/nist-mud.git
```



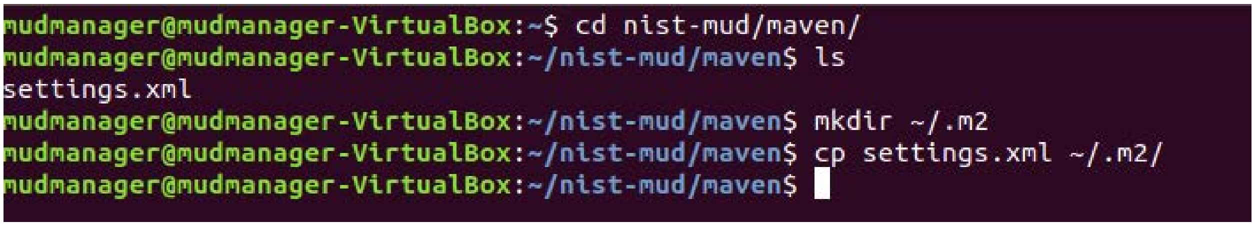
```
mudmanager@mudmanager-VirtualBox:~$ git clone https://github.com/usnistgov/nist
-mud.git
```

2. Copy the contents of `nist-mud/maven/settings.xml` to `~/.m2` by executing the commands below:

```
cd nist-mud/maven/
```

```
mkdir ~/.m2
```

```
cp settings.xml ~/.m2
```



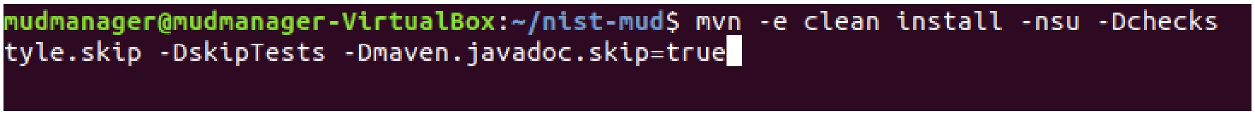
```
mudmanager@mudmanager-VirtualBox:~$ cd nist-mud/maven/
mudmanager@mudmanager-VirtualBox:~/nist-mud/maven$ ls
settings.xml
mudmanager@mudmanager-VirtualBox:~/nist-mud/maven$ mkdir ~/.m2
mudmanager@mudmanager-VirtualBox:~/nist-mud/maven$ cp settings.xml ~/.m2/
mudmanager@mudmanager-VirtualBox:~/nist-mud/maven$
```

3. In the nist-mud directory, run the commands below:

```
cd
```

```
cd nist-mud/
```

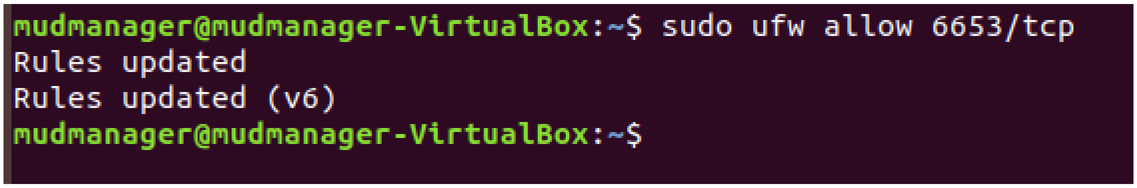
```
mvn -e clean install -nsu -Dcheckstyle.skip -DskipTests -
Dmaven.javadoc.skip=true
```



```
mudmanager@mudmanager-VirtualBox:~/nist-mud$ mvn -e clean install -nsu -Dchecks
tyle.skip -DskipTests -Dmaven.javadoc.skip=true
```

4. Open port 6653 on the controller stack for TCP access so the switches can connect by executing the command below:

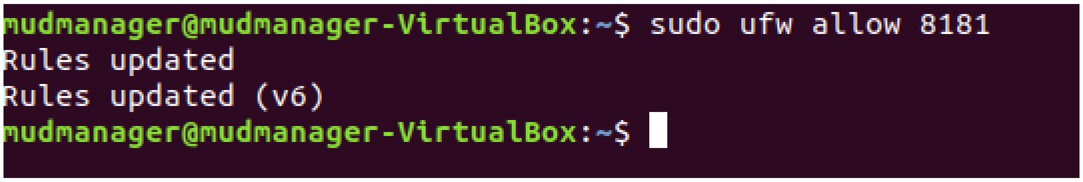
```
sudo ufw allow 6653/tcp
```



```
mudmanager@mudmanager-VirtualBox:~$ sudo ufw allow 6653/tcp
Rules updated
Rules updated (v6)
mudmanager@mudmanager-VirtualBox:~$
```

5. OpenDaylight uses port 8181 for the REST API. That port should be opened if access to the REST API is desired from outside the controller machine. Open port 8181 by executing the command below:

```
sudo ufw allow 8181
```



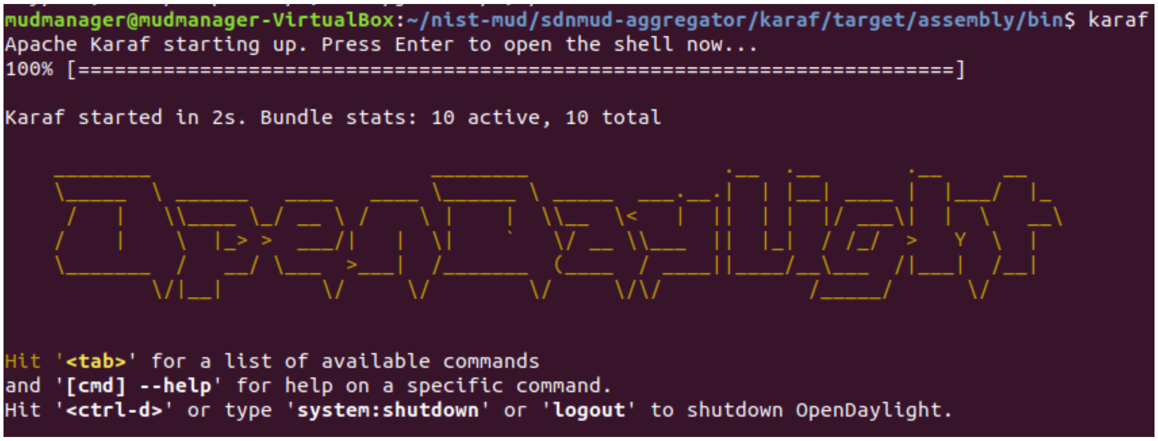
```
mudmanager@mudmanager-VirtualBox:~$ sudo ufw allow 8181
Rules updated
Rules updated (v6)
mudmanager@mudmanager-VirtualBox:~$
```

6. Change to the bin directory by executing the command below:

   `~/nist-mud/sdnmud-aggregator/karaf/target/assembly/bin`

7. Run the command below:

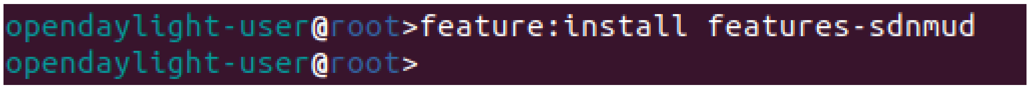
   `./karaf clean`



```
mudmanager@mudmanager-VirtualBox:~/nist-mud/sdnmud-aggregator/karaf/target/assembly/bin$ karaf
Apache Karaf starting up. Press Enter to open the shell now...
100% [========================================================================]

Karaf started in 2s. Bundle stats: 10 active, 10 total

    ________                       ________                .__  .__       .__     __
    \_____  \ ______   ____   ____ \______ \ _____  ___.__.|  | |__| ____ |  |___/  |_
     /   |   \\____ \_/ __ \ /    \ |    |  \\__  \<   |  ||  | |  |/ ___\|  |  \   __\
    /    |    \  |_> >  ___/|   |  \|    `   \/ __ \\___  ||  |_|  / /_/  >   Y  \  |
    \_______  /   __/ \___  >___|  /_______  (____  / ____||____/__\___  /|___|  /__|
            \/|__|        \/     \/        \/     \/\/            /_____/      \/


Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

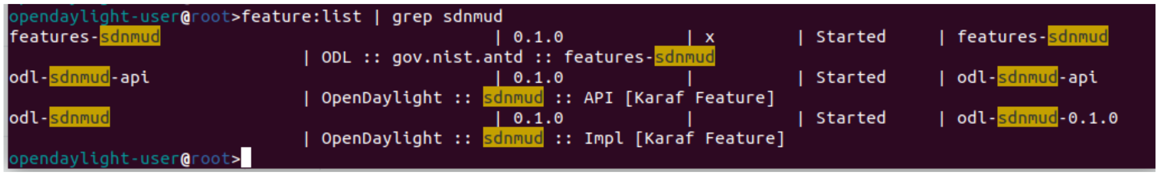
8. At the Karaf prompt, install MUD capabilities using:

   `feature:install features-sdnmud`



```
opendaylight-user@root>feature:install features-sdnmud
opendaylight-user@root>
```

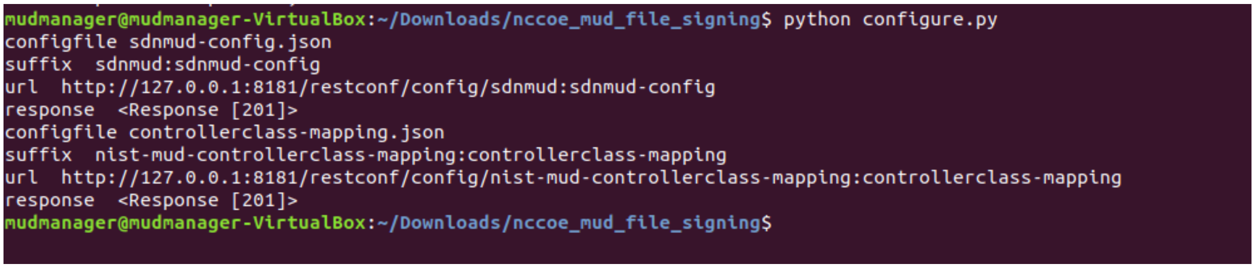
9. Check if the feature is running by using the command `feature:list | grep sdnmud` in Karaf.



```
opendaylight-user@root>feature:list | grep sdnmud
features-sdnmud                                  | 0.1.0            | x        | Started     | features-sdnmud
                              | ODL :: gov.nist.antd :: features-sdnmud
odl-sdnmud-api                                   | 0.1.0            |          | Started     | odl-sdnmud-api
                              | OpenDaylight :: sdnmud :: API [Karaf Feature]
odl-sdnmud                                       | 0.1.0            |          | Started     | odl-sdnmud-0.1.0
                              | OpenDaylight :: sdnmud :: Impl [Karaf Feature]
opendaylight-user@root>
```

10. On the SDN controller/MUD manager host, run a script to configure the SDN controller and add bindings for the controller abstractions defined in the test MUD files. This script pushes configuration information for the MUD manager application (`sdnmud-config.json`) as well as network configuration information for the managed local area network (LAN) (`controllerclass-mapping.json`). The latter file specifies bindings for the controller classes that are used in the MUD

file as well as subnet information for classification of local addresses. These are scoped to a single policy enforcement point, which is identified by a switch-id. By default, the switch ID is `openflow:MAC-address` where `MAC-address` is the MAC address of the switch interface that connects to the SDN controller (in decimal). This must be unique per switch. Note too, that we identify whether a switch is wireless.



```
mudmanager@mudmanager-VirtualBox:~/Downloads/nccoe_mud_file_signing$ python configure.py
configfile sdnmud-config.json
suffix  sdnmud:sdnmud-config
url  http://127.0.0.1:8181/restconf/config/sdnmud:sdnmud-config
response  <Response [201]>
configfile controllerclass-mapping.json
suffix  nist-mud-controllerclass-mapping:controllerclass-mapping
url  http://127.0.0.1:8181/restconf/config/nist-mud-controllerclass-mapping:controllerclass-mapping
response  <Response [201]>
mudmanager@mudmanager-VirtualBox:~/Downloads/nccoe_mud_file_signing$
```

Example Python script (`configure.py`):

```
import requests
import json
import argparse
import os

if __name__=="__main__":
   if os.environ.get("CONTROLLER_ADDR") is None:
      print "Please set environment variable CONTROLLER_ADDR to the address of the 
opendaylight controller"

   controller_addr = os.environ.get("CONTROLLER_ADDR")

   headers= {"Content-Type":"application/json"}
   for (configfile,suffix) in {
       ("sdnmud-config.json", "sdnmud:sdnmud-config"),
       ("controllerclass-mapping.json","nist-mud-controllerclass-
mapping:controllerclass-mapping") }:
       data = json.load(open(configfile))
       print "configfile", configfile
       print "suffix ", suffix
       url = "http://" + controller_addr + ":8181/restconf/config/" + suffix
       print "url ", url
       r = requests.put(url, data=json.dumps(data), headers=headers , auth=('admin', 
'admin'))
       print "response ", r
```

Example controller class mapping (`controllerclass-mapping.json`):

```
{
"controllerclass-mapping" :  {
   "switch-id" : "openflow:123917682138002",
   "controller" : [
       {
             "uri" :  "urn:ietf:params:mud:dns",
             "address-list" : [ "10.0.41.1" ]
       },
       {
              "uri" :  "urn:ietf:params:mud:dhcp",
              "address-list" : [ "10.0.41.1" ]
       },
       {
              "uri" :  "https://controller.nist.local",
              "address-list" : [ "10.0.41.225" ]
       },
       {
              "uri" :  "https://sensor.nist.local/nistmud1",
              "address-list" : [ "10.0.41.225" ]
       }
   ],
   "local-networks": [ "10.0.41.0/24" ],
   "wireless" : true
 }
}
```

Example SDN MUD configuration (`sdnmud-config.json`):

```
{
 "sdnmud-config" : {
       "ca-certs": "lib/security/cacerts",
       "key-pass" : "changeit",
       "trust-self-signed-cert" : true,
       "mfg-id-rule-cache-timeout": 120,
       "relaxed-acl" : false
  }
}
```

5.2 MUD File Server

5.2.1 MUD File Sever Overview

The MUD file server is responsible for serving the MUD file and the corresponding signature file upon request from the MUD manager. For testing purposes, the MUD file server is run on 127.0.0.1 on the same machine as the MUD manager. This allows us to examine the logs to check if the MUD file has been retrieved. For testing purposes, host name verification for the TLS connection to the MUD file server is disabled in the configuration of the MUD manager.

5.2.2 Configuration Overview

The following subsections document the software, hardware, and network configurations for the MUD file server.

5.2.2.1 Hardware Configuration

The MUD file server was hosted on the same machine as the SDN controller.

5.2.2.2 Network Configuration

The MUD file server was hosted on the same machine as the SDN controller. To direct the MUD manager to retrieve the MUD files from the MUD file server, the host name of the two manufacturers that are present in the MUD URLs used for testing are both mapped to 127.0.0.1 in the */etc/hosts* file of the Java Virtual Machine in which the MUD manager is running. This static configuration is read by the MUD manager when it starts. The name resolution information in the */etc/hosts* file directs the MUD manager to retrieve the test MUD files from the MUD file server.

5.2.2.3 Software Configuration

In this build, serving MUD files requires Python 2.7 and the Python requests package. These may be installed using *apt* and *pip*. After creation of the MUD files by using mudmaker.org, the MUD files were signed, and the certificates used for signing were imported into the trust store of the Java Virtual Machine in which the MUD manager is running.

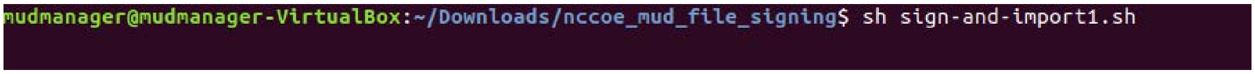
5.2.3 Setup

5.2.3.1 MUD File Creation

This build also leveraged the MUD Maker online tool found at www.mudmaker.org. For detailed instructions on creating a MUD file using this online tool, please refer to Build 1's MUD File Creation section.

5.2.3.2 MUD File Signing

1. Sign and import the desired MUD files. An example script (`sign-and-import1.sh`) can be found below.



```
mudmanager@mudmanager-VirtualBox:~/Downloads/nccoe_mud_file_signing$ sh sign-and-import1.sh
```

The shell script that was used in this build is shown below. This script generates a signature based on the private key of a DigiCert-issued certificate and imports the certificate into the trust store of the Java Virtual Machine. This is done for both MUD files.

```
CACERT=DigiCertCA.crt
MANUFACTURER_CRT=nccoe_mud_file_signing.crt
MANUFACTURER_KEY=mudsign.key.pem
MANUFACTURER_ALIAS=sensor.nist.local
MANUFACTURER_SIGNATURE=mudfile-sensor.p7s
MUDFILE=mudfile-sensor.json

openssl cms -sign -signer $MANUFACTURER_CRT -inkey $MANUFACTURER_KEY -in $MUDFILE -
binary -noattr -outform DER -certfile $CACERT  -out $MANUFACTURER_SIGNATURE
openssl cms -verify -binary  -in $MANUFACTURER_SIGNATURE  -signer $MANUFACTURER_CRT -
inform DER  -content $MUDFILE
```

```
MANUFACTURER_ALIAS=otherman.nist.local
MUDFILE=mudfile-otherman.json
MANUFACTURER_SIGNATURE=mudfile-otherman.p7s
openssl cms -sign -signer $MANUFACTURER_CRT -inkey $MANUFACTURER_KEY -in $MUDFILE -
binary -noattr -outform DER -certfile $CACERT  -out $MANUFACTURER_SIGNATURE
openssl cms -verify -binary  -in $MANUFACTURER_SIGNATURE  -signer $MANUFACTURER_CRT -
inform DER  -content $MUDFILE

sudo -E $JAVA_HOME/bin/keytool -delete -alias digicert -keystore
$JAVA_HOME/jre/lib/security/cacerts -storepass changeit
sudo -E $JAVA_HOME/bin/keytool -importcert -file $CACERT -alias digicert -keystore
$JAVA_HOME/jre/lib/security/cacerts -storepass changeit
```

5.2.3.3 MUD File Serving

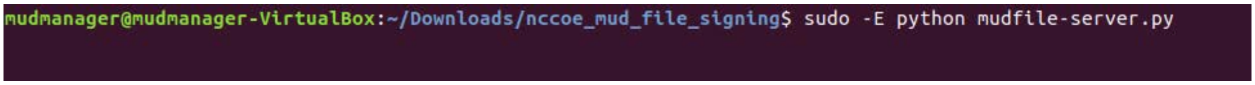
Run a script that serves desired MUD files and signatures. An example Python script (`mudfile-server.py`) can be found below.

1. Save a copy of the **mudfile-server.py** Python script onto the NIST SDN controller/MUD manager configured in Section 5.1:

```
import BaseHTTPServer, SimpleHTTPServer
import ssl
import urlparse
# Dummy manufacturer server for testing

class MyHTTPRequestHandler(SimpleHTTPServer.SimpleHTTPRequestHandler):

    def do_GET(self):
        print ("DoGET " + self.path)
        self.send_response(200)
        if self.path == "/nistmud1" :
           with  open("mudfile-sensor.json", mode="r") as f:
                data = f.read()
          print("Read " + str(len(data)) + " chars ")
                self.send_header("Content-Length", len(data))
                self.end_headers()
                self.wfile.write(data)
        elif self.path == "/nistmud2" :
           with  open("mudfile-otherman.json", mode="r") as f:
                data = f.read()
          print("Read " + str(len(data)) + " chars ")
                self.send_header("Content-Length", len(data))
                self.end_headers()
                self.wfile.write(data)
        elif self.path == "/nistmud1/mudfile-sensor.p7s":
           with open("mudfile-sensor.p7s",mode="r") as f:
                data = f.read()
          print("Read " + str(len(data)) + " chars ")
                self.send_header("Content-Length", len(data))
                self.end_headers()
                self.wfile.write(data)
        elif self.path == "/nistmud2/mudfile-otherman.p7s":
           with open("mudfile-otherman.p7s",mode="r") as f:
                data = f.read()
          print("Read " + str(len(data)) + " chars ")
                self.send_header("Content-Length", len(data))
                self.end_headers()
                self.wfile.write(data)
        else:
           print("UNKNOWN URL!!")
           self.wfile.write(b'Hello, world!')

httpd = BaseHTTPServer.HTTPServer(('0.0.0.0', 443), MyHTTPRequestHandler)
httpd.socket = ssl.wrap_socket (httpd.socket, keyfile='./mudsigner.key',
certfile='./mudsigner.crt', server_side=True)
httpd.serve_forever()
```

2. From the same directory as the previous step, execute the command below to start the MUD file server:

   ```
   sudo -E python mudfile-server.py
   ```



```
mudmanager@mudmanager-VirtualBox:~/Downloads/nccoe_mud_file_signing$ sudo -E python mudfile-server.py
```

5.3 Northbound Networks Zodiac WX Access Point

5.3.1 Northbound Networks Zodiac WX Access Point Overview

The Zodiac WX, in addition to being a wireless access point, includes the following logical components: an SDN switch, a NAT router, a DHCP server, and a DNS server. The Zodiac WX is powered by OpenWRT and Open vSwitch. Open vSwitch directly integrates into the wireless configuration. The Zodiac WX works with any standard OpenFlow-compatible controllers and requires no modifications because it appears to the controller as a standard OpenFlow switch.

5.3.2 Configuration Overview

The following subsections document the network, software, and hardware configurations for the SDN-capable Northbound Networks Zodiac WX.

5.3.2.1 Network Configuration

The access point is configured to have a static public address on the public side of the NAT. For purposes of testing, we use 203.0.113.x addresses on the public network. The public side of the NAT is given the address of 203.0.113.1. The DHCP server is set up to allocate addresses to wireless devices on the LAN. The SDN controller/MUD manager is connected to the public side of the NAT. The Open vSwitch configuration for the access point is given the address of the SDN controller, which is shown in the setup below.

5.3.2.2 Software Configuration

At this implementation, no additional software configuration was required.

5.3.2.3 Hardware Configuration

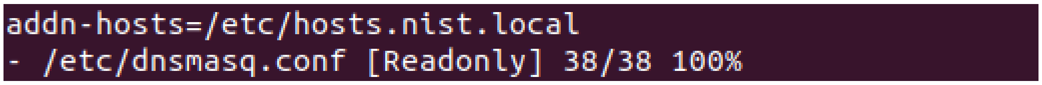
At this implementation, no additional hardware configuration was required.

5.3.3 Setup

On the Zodiac WX, DNSmasq supports both DHCP and DNS. For testing purposes, it will be necessary to access several web servers (two update servers called www.nist.local and an unapproved server called www.antd.local). The following commands enable the Zodiac WX to resolve the web server host names to their IP addresses.

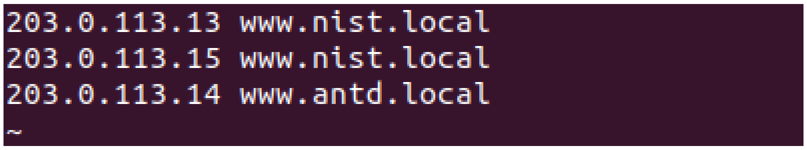
1. Set up the access point to resolve the addresses for the web server host names by opening the file `/etc/dnsmasq.conf` on the access point.
2. Add the following line to the *`dnsmasq.conf`* file:

   ```
   addn-hosts=/etc/hosts.nist.local
   ```

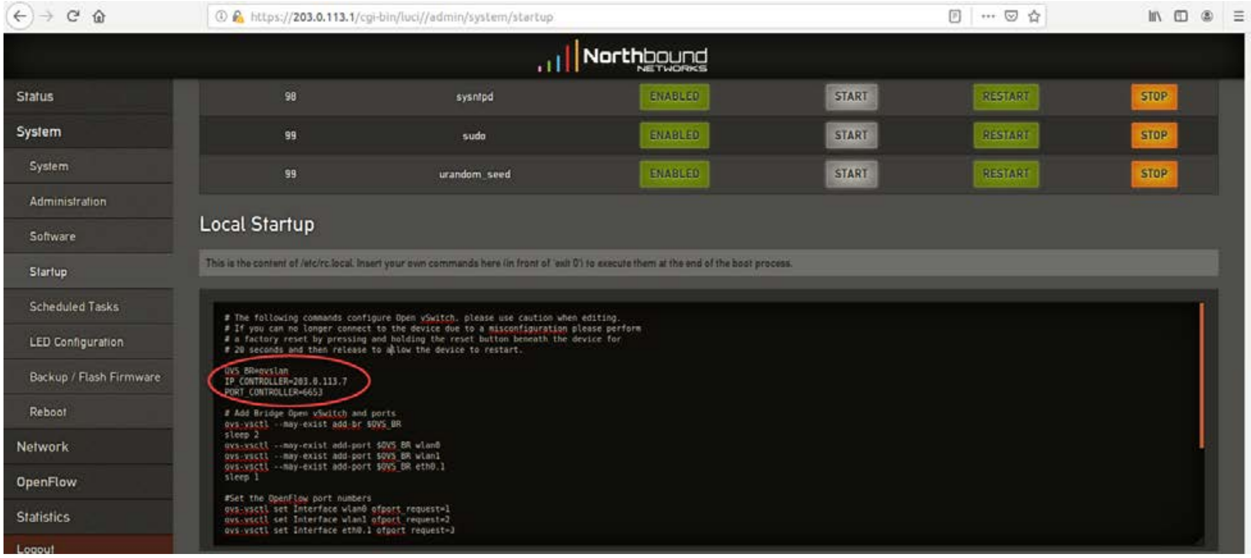
   ```
   addn-hosts=/etc/hosts.nist.local
   - /etc/dnsmasq.conf [Readonly] 38/38 100%
   ```

3. The file */etc/hosts.nist.local* has the host name to address mapping. The mapping used for our tests is shown below (Note that the host www.nist.local maps to two addresses on the public side).

   ```
   203.0.113.13 www.nist.local
   203.0.113.15 www.nist.local
   203.0.113.14 www.antd.local
   ~
   ```

4. On the Zodiac WX configuration web page in the System->Startup tab, indicate where (IP address and port) the Open vSwitch Daemon connects to the controller.



98	sysntpd	ENABLED	START	RESTART	STOP
99	sudo	ENABLED	START	RESTART	STOP
99	urandom_seed	ENABLED	START	RESTART	STOP

Local Startup

This is the content of /etc/rc.local. Insert your own commands here (in front of 'exit 0') to execute them at the end of the boot process.

```
# The following commands configure Open vSwitch. please use caution when editing.
# If you can no longer connect to the device due to a misconfiguration please perform
# a factory reset by pressing and holding the reset button beneath the device for
# 20 seconds and then release to allow the device to restart.

OVS_BR=ovslan
IP_CONTROLLER=203.0.113.7
PORT_CONTROLLER=6653

# Add Bridge Open vSwitch and ports
ovs-vsctl --may-exist add-br $OVS_BR
sleep 2
ovs-vsctl --may-exist add-port $OVS_BR wlan0
ovs-vsctl --may-exist add-port $OVS_BR wlan1
ovs-vsctl --may-exist add-port $OVS_BR eth0.1
sleep 1

#Set the OpenFlow port numbers
ovs-vsctl set Interface wlan0 ofport_request=1
ovs-vsctl set Interface wlan1 ofport_request=2
ovs-vsctl set Interface eth0.1 ofport_request=3
```

5.4 DigiCert Certificates

DigiCert's CertCentral web-based platform allows provisioning and management of publicly trusted X.509 certificates for a variety of purposes. After establishing an account, clients can log in, request, renew, and revoke certificates by using only a browser. For Build 4, the Premium Certificate created in Build 1 was leveraged for signing the MUD files. To request and implement DigiCert certificates, follow the documentation in Build 1's DigiCert Certificates section and subsequent sections.

5.5 IoT Devices

5.5.1 IoT Devices Overview

This section provides configuration details for the Linux-based Raspberry Pis used in the build, which emit MUD URLs by using DHCP.

5.5.2 Configuration Overview

The devices used in this build were multiple Raspberry Pi development kits that were configured to act as IoT devices. The devices run Raspbian 9, a Linux-based operating system, and are configured to emit a MUD URL during a typical DHCP transaction. These devices were used to test interactions related to MUD capabilities.

5.5.2.1 Network Configuration

The kits are connected to the network over a wireless connection. Their IP addresses are assigned dynamically by the DHCP server on the Zodiac WX access point.

5.5.2.2 Software Configuration

The Raspberry Pis are configured on Raspbian. They also utilized dhclient as their default DHCP clients to manually initiate a DHCP interaction. This DHCP client is installed natively on many Linux distributions and can be installed using a preferred package manager if not currently present. Dhclient uses a configuration file: `/etc/dhclient.conf`. This needs to be modified to include the MUD URL that the device will emit in its DHCP requests. (The modification details are provided in the setup information below.)

5.5.2.3 Hardware Configuration

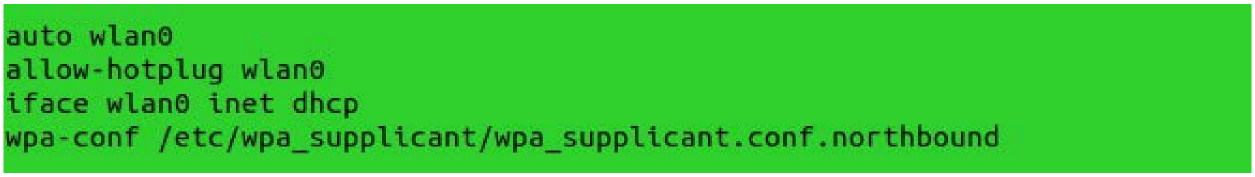
Multiple Raspberry Pi 3 Model B devices were used.

5.5.3 Setup

Each Raspberry Pi used in this build was intended to represent a different class of device (manufacturer, other manufacturer, local networks, controller classes). The type of device was determined by the MUD URL being emitted by the device. If no MUD URL is emitted, the device is an unclassified local network device.

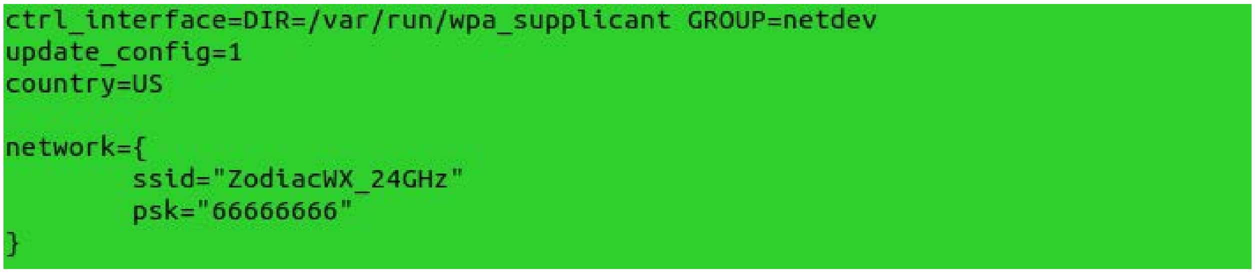
1. On each Pi, changes were made to `/etc/network/interfaces` to add a line that allows the Pi to authenticate to the access point. The following line is added to the network interface as shown below:

   ```
   wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf.northbound
   ```

   ```
   auto wlan0
   allow-hotplug wlan0
   iface wlan0 inet dhcp
   wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf.northbound
   ```

   The file (*/etc/wpa_supplicant/wpa_supplicant.conf.northbound*) is shown below:

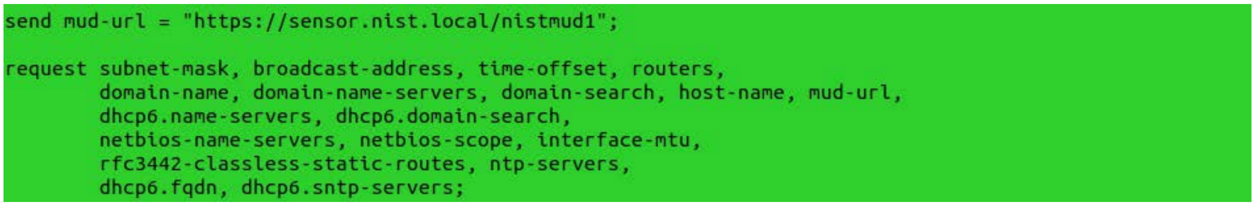
   ```
   ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
   update_config=1
   country=US

   network={
           ssid="ZodiacWX_24GHz"
           psk="66666666"
   }
   ```

2. A dhclient configuration file can be altered (by adding information) to allow for emission of a MUD URL in the DHCP transaction. Modify the *dhclient.conf* file with the command:

```
vi /etc/dhcp/dhclient.conf
```

3. A send MUD URL line must be added as well as a `mud-url` in the request line. In this build, multiple MUD URLs were transmitted, depending on the type of the device. Example alterations made to dhclient configuration files can be seen below:

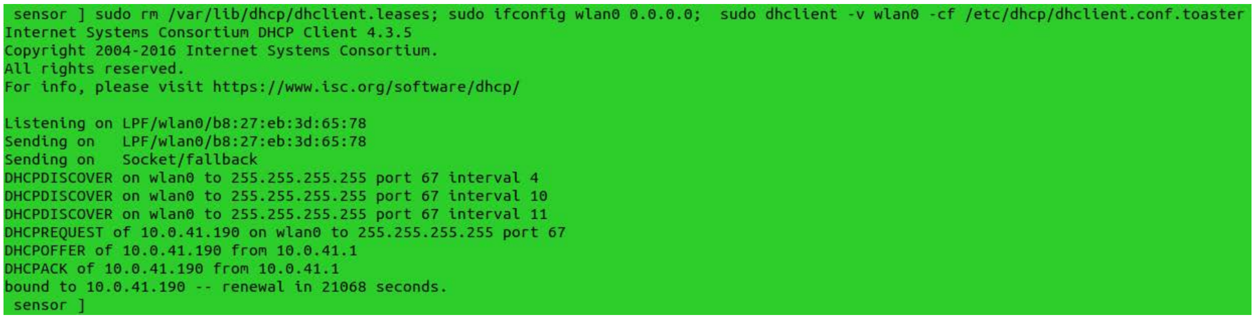
```
send mud-url = "https://sensor.nist.local/nistmud1";
send mud-url = "https://otherman.nist.local/nistmud2";
```



```
send mud-url = "https://sensor.nist.local/nistmud1";

request subnet-mask, broadcast-address, time-offset, routers,
        domain-name, domain-name-servers, domain-search, host-name, mud-url,
        dhcp6.name-servers, dhcp6.domain-search,
        netbios-name-servers, netbios-scope, interface-mtu,
        rfc3442-classless-static-routes, ntp-servers,
        dhcp6.fqdn, dhcp6.sntp-servers;
```

4. To control the time at which the MUD URL is emitted, we manually reacquire the DHCP address rather than have the device acquire the MUD URL on boot. Emit the MUD URL and attain an IP address by sending the altered dhclient configuration file manually with the following commands:

```
sudo rm /var/lib/dhcp/dhclient.leases
sudo ifconfig wlan0 0.0.0.0
sudo dhclient -v wlan0 -cf /etc/dhcp/dhclient.conf.toaster
```



```
 sensor ] sudo rm /var/lib/dhcp/dhclient.leases; sudo ifconfig wlan0 0.0.0.0;  sudo dhclient -v wlan0 -cf /etc/dhcp/dhclient.conf.toaster
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/b8:27:eb:3d:65:78
Sending on   LPF/wlan0/b8:27:eb:3d:65:78
Sending on   Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 4
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 10
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 11
DHCPREQUEST of 10.0.41.190 on wlan0 to 255.255.255.255 port 67
DHCPOFFER of 10.0.41.190 from 10.0.41.1
DHCPACK of 10.0.41.190 from 10.0.41.1
bound to 10.0.41.190 -- renewal in 21068 seconds.
 sensor ]
```

5.6 Update Server

5.6.1 Update Server Overview

This section provides configuration details for the Linux-based IoT development kit used in the build, which acts as an update server. This update server will attempt to access and be accessed by the IoT device, which, in this case, is one of the development kits built in the lab. The update server is a web server that hosts mock software update files to be served as software updates to our IoT device devkits. When the server receives an http request, it sends the corresponding update file.

5.6.2 Configuration Overview

The devkit runs Raspbian 9, a Linux-based operating system, and is configured to act as an update server. This host was used to test approved internet interactions related to MUD capabilities.

5.6.2.1 Network Configuration

The web server host has a static public IP address configuration and is connected to the access point on the wired interface. It is given an address on the 203.0.113 network.

5.6.2.2 Software Configuration

The Raspberry Pi is configured on Raspbian. The devkit also utilized a simple Python script to run an http server to test MUD capabilities.

5.6.2.3 Hardware Configuration

The hardware used for this devkit includes a Raspberry Pi 3 Model B.

5.6.3 Setup

The primary configuration needed for the web server device is done with the DNS mapping on the Zodiac WX access point to be discussed in the section related to setup of the Northbound Networks Zodiac WX Access Point. The Raspberry Pi is required to run a simple http server.

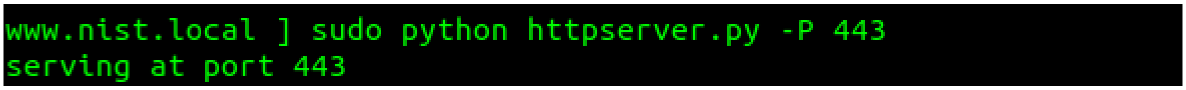
1. Copy the example Python script below onto the Raspberry Pi:

   Example Python script (`httpserver.py`):

   ```
   import SimpleHTTPServer
   import SocketServer
   import argparse
   if __name__ == "__main__":
       parser = argparse.ArgumentParser()
       parser.add_argument("-H", help="Host address", default="0.0.0.0")
       parser.add_argument("-P", help="Port ", default="80")
       args = parser.parse_args()
       hostAddr = args.H
       PORT = int(args.P)
       Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
       httpd = SocketServer.TCPServer((hostAddr, PORT), Handler)
       print "serving at port", PORT
       httpd.serve_forever()
   ```

2. From the same directory as the script copied in the previous step, execute the command below to start the http server:

```
sudo python httpserver.py -P 443
```



```
www.nist.local ] sudo python httpserver.py -P 443
serving at port 443
```

5.7 Unapproved Server

5.7.1 Unapproved Server Overview

This section provides configuration details for the Linux-based IoT development kit used in the build, which acts as an unapproved internet host. This host will attempt to access and to be accessed by an IoT device, which, in this case, is one of the MUD-capable devices on the network.

The unapproved server is an internet host that is not explicitly authorized in the MUD file to communicate with the IoT device. When the IoT device attempts to connect to this server, the switch should not allow this traffic because it is not an approved internet service per the corresponding MUD file. Likewise, when the server attempts to connect to the IoT device, this traffic should be denied at the switch.

5.7.2 Configuration Overview

The devkit runs Raspbian 9, a Linux-based operating system, and is configured to act as an unapproved internet host. This host was used to test unapproved internet interactions related to MUD capabilities.

5.7.2.1 Network Configuration

The web host has a static public IP address configuration and is connected to the access point on the wired interface. It is given an address on the 203.0.113 network.

5.7.2.2 Software Configuration

The Raspberry Pi is configured on Raspbian. The devkit also utilized a simple Python script to run an http server to test MUD capabilities.

5.7.2.3 Hardware Configuration

The hardware used for this devkit includes a Raspberry Pi 3 Model B.

5.7.3 Setup

The primary configuration needed for the web server device is accomplished by the DNS mapping on the Zodiac WX access point to be discussed in the section related to setup of the Northbound Networks Zodiac WX Access Point. The Raspberry Pi is required to run a simple http server.

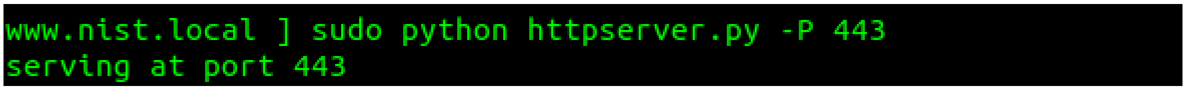
1. Copy the example Python script below onto the Raspberry Pi:

Example Python script (`httpserver.py`):

```
import SimpleHTTPServer
import SocketServer
import argparse
if __name__ == "__main__":
   parser = argparse.ArgumentParser()
   parser.add_argument("-H", help="Host address", default="0.0.0.0")
   parser.add_argument("-P", help="Port ", default="80")
   args = parser.parse_args()
   hostAddr = args.H
   PORT = int(args.P)
   Handler = SimpleHTTPServer.SimpleHTTPRequestHandler
   httpd = SocketServer.TCPServer((hostAddr, PORT), Handler)
   print "serving at port", PORT
   httpd.serve_forever()
```

2. From the same directory as the script copied in the previous step, execute the command below to start the http server:

```
sudo python httpserver.py -P 443
```



```
www.nist.local ] sudo python httpserver.py -P 443
serving at port 443
```

3831 Appendix A List of Acronyms

AAA	Authentication, Authorization, and Accounting
ACL	Access Control List
API	Application Programming Interface
CMS	Cryptographic Message Syntax
COA	Change of Authorization
CRADA	Cooperative Research and Development Agreement
DB	Database
DDoS	Distributed Denial of Service
Devkit	Development Kit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GCA	Global Cyber Alliance
http	Hypertext Transfer Protocol
https	Hypertext Transfer Protocol Secure
IOS	Cisco's Internetwork Operating System
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IT	Information Technology
ITL	NIST's Information Technology Laboratory
JSON	JavaScript Object Notation
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LED	Light-Emitting Diode

LLDP	Link Layer Discovery Protocol (**Institute of Electrical and Electronics Engineers** 802.1AB)
MAB	MAC Authentication Bypass
MAC	Media Access Control
MQTT	Message Queuing Telemetry Transport
MUD	Manufacturer Usage Description
NAS	Network Access Server
NAT	Network Address Translation
NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
OS	Operating System
PoE	Power over Ethernet
RADIUS	Remote Authentication Dial-In User Service
REST	Representational State Transfer
RFC	Request for Comments
SDN	Software-Defined Networking
SP	Special Publication
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UI	User Interface
URL	Uniform Resource Locator
Vi	Visual
VLAN	Virtual Local Area Network

VNC	Virtual Network Computing
WAN	Wide Area Network

Appendix B Glossary

Audit

Independent review and examination of records and activities to assess the adequacy of system controls to ensure compliance with established policies and operational procedures (National Institute of Standards and Technology [NIST] Special Publication [SP] 800-12 Rev. 1)

Best Practice

A procedure that has been shown by research and experience to produce optimal results and that is established or proposed as a standard suitable for widespread adoption (Merriam-Webster)

Botnet

The word “botnet” is formed from the words “robot” and “network.” Cybercriminals use special Trojan viruses to breach the security of several users’ computers, take control of each computer, and organise all of the infected machines into a network of “bots” that the criminal can remotely manage. (https://usa.kaspersky.com/resource-center/threats/botnet-attacks)

Control

A measure that is modifying risk (Note: Controls include any process, policy, device, practice, or other actions that modify risk.) (NIST Interagency or Internal Report 8053)

Denial of Service

The prevention of authorized access to a system resource or the delaying of system operations and functions (NIST SP 800-82 Rev. 2)

Distributed Denial of Service (DDoS)

A denial of service technique that uses numerous hosts to perform the attack (NIST Interagency or Internal Report 7711)

Managed Devices

Personal computers, laptops, mobile devices, virtual machines, and infrastructure components require management agents, allowing information technology staff to discover, maintain, and control these devices. Those with broken or missing agents cannot be seen or managed by agent-based security products.

Manufacturer Usage Description (MUD)

A component-based architecture specified in Request for Comments (RFC) 8250 that is designed to provide a means for end devices to signal to the network what sort of access and network functionality they require to properly function

Mapping

Depiction of how data from one information source maps to data from another information source

Mitigate To make less severe or painful or to cause to become less harsh or hostile (Merriam-Webster)

MUD-Capable An IoT device that is capable of emitting a MUD uniform resource locator (URL) in compliance with the MUD specification

Network Address Translation (NAT) A function by which internet protocol (IP) addresses within a packet are replaced with different IP addresses. This function is most commonly performed by either **routers** or firewalls. It enables private IP networks that **use** unregistered IP addresses to connect to the internet. **NAT** operates on a router, usually connecting two networks together, and translates the private (not globally unique) addresses in the internal network into legal addresses before packets are forwarded to another network.

Non-MUD-Capable An IoT device that is not capable of emitting a MUD URL in compliance with the MUD specification (RFC 8250)

Policy Statements, rules, or assertions that specify the correct or expected behavior of an entity. For example, an authorization policy might specify the correct access control rules for a software component. (NIST SP 800-95 and NIST Interagency or Internal Report 7621 Rev. 1)

Policy Enforcement Point A network device on which policy decisions are carried out or enforced

Risk The net negative impact of the exercise of a vulnerability, considering both the probability and the impact of occurrence. Risk management is the process of identifying risk, assessing risk, and taking steps to reduce risk to an acceptable level. (NIST SP 800-30)

Router A computer that is a gateway between two networks at open systems interconnection layer 3 and that relays and directs data packets through that internetwork. The most common form of router operates on IP packets. (NIST SP 800-82 Rev. 2)

Security Control A safeguard or countermeasure prescribed for an information system or an organization, which is designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements (NIST SP 800-53 Rev. 4)

Server A computer or device on a network that manages network resources. Examples are file servers (to store files), print servers (to manage one or more printers), network servers (to manage network traffic), and database servers (to process database queries). (NIST SP 800-47)

Shall A requirement that must be met unless a justification of why it cannot be met is given and accepted (NIST Interagency or Internal Report 5153)

Should This term is used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. (NIST SP 800-108)

Threat Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat source to successfully exploit a particular information system vulnerability (Federal Information Processing Standards 200)

Threat Signaling Real-time signaling of DDoS-related telemetry and threat-handling requests and data between elements concerned with DDoS attack detection, classification, traceback, and mitigation (https://joinup.ec.europa.eu/collection/rolling-plan-ict-standardisation/cybersecurity-network-and-information-security)

Traffic Filter An entry in an access control list that is installed on the router or switch to enforce access controls on the network

Uniform Resource Locator (URL) A reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A typical URL could have the form http://www.example.com/index.html, which indicates a protocol (hypertext transfer protocol [http]), a host name (www.example.com), and a file name *(index.html)*. Also sometimes referred to as a *web address*

Update New, improved, or fixed software, which replaces older versions of the same software. For example, updating an OS brings it up-to-date with the latest drivers, system utilities, and security software. Updates are often provided by the software publisher free of charge. (https://www.computerhope.com/jargon/u/update.htm)

Update Server A server that provides patches and other software updates to Internet of Things devices

Virtual Local Area Network (VLAN) A broadcast domain that is partitioned and isolated within a network at the data link layer. A single physical local area network (LAN) can be logically partitioned into multiple, independent VLANs; a group of devices on one or more physical LANs can be configured to communicate within the same VLAN as if they were attached to the same physical LAN.

Vulnerability Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source (NIST SP 800-37 Rev. 2)

Appendix C Bibliography

Request for Comments (RFC) 8520. (2019, Mar.) “Manufacturer Usage Description Specification” [Online]. Available: https://tools.ietf.org/html/rfc8520.

Cisco’s developer MUD Manager GitHub page [Website]. Available: https://github.com/CiscoDevNet/MUD-Manager/tree/1.0#dependancies.

Apache HTTP Server Project documentation, Version 2.4. Compiling and Installing Apache [Website]. Available: https://httpd.apache.org/docs/current/install.html.

Apache HTTP Server Project documentation, Version 2.4. Apache SSL/TLS Encryption [Website]. Available: https://httpd.apache.org/docs/current/ssl/ssl_howto.html.

Welcome to MUD File maker! [Website]. Available: https://www.mudmaker.org/.

DigiCert. Advanced CertCentral Getting Started Guide, Version 9.2 [Website]. Available: https://www.digicert.com/certcentral-support/digicert-getting-started-guide.pdf.

DigiCert. SSL Certificate Support [Website]. Available: https://www.digicert.com/security-certificate-support/.

DigiCert. Order your SSL/TLS certificates [Website]. Available: https://docs.digicert.com/manage-certificates/order-your-ssltls-certificates/.

DigiCert. CertCentral Client Certificate Guide, Version 1.9 [Website]. Available: https://www.digicert.com/certcentral-support/client-certificate-guide.pdf.

Forescout. ForeScout CounterAct® Installation Guide, Version 8.0.1 [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/10/CounterACT_Installation_Guide_8.0.1.pdf.

Forescout. (2018, Feb.) ForeScout CounterAct Device Profile Library Configuration Guide [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Device_Profile_Library.pdf.

Forescout. (2018, Feb.) ForeScout CounterAct IoT Posture Assessment Library Configuration Guide [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_IoT_Posture_Assessment_Library-1.pdf.

Forescout. ForeScout CounterAct Open Integration Module Overview Guide, Version 1.1 [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/08/CounterACT_Open_Integration_Module_Overview_1.1.pdf.

Forescout. (2018, Feb.) ForeScout CounterAct Windows Applications Configuration Guide [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Applications.pdf.

Forescout. (2018, Feb.) ForeScout CounterAct Windows Vulnerability DB Configuration Guide [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/04/CounterACT_Windows_Vulnerability_DB_18.0.2.pdf.

Forescout. HPS NIC Vendor DB Configuration Guide, Version 1.2.4 [Website]. Available: https://www.Forescout.com/wp-content/uploads/2018/04/HPS_NIC_Vendor_DB_1.2.4.pdf.