NIST SPECIAL PUBLICATION 1800-36C

Trusted Internet of Things (IoT) Device Network-Layer Onboarding and Lifecycle Management:

Enhancing Internet Protocol-Based IoT Device and Network Security

Volume C:
How-To Guides

Murugiah Souppaya
Paul Watrobski
National Institute of Standards and Technology
Gaithersburg, Maryland

Chelsea Deane
Joshua Klosterman
Blaine Mulugeta
Charlie Rearick
Susan Symington
The MITRE Corporation
McLean, Virginia

Dan Harkins
Danny Jump
Aruba, a Hewlett Packard
Enterprise Company
San Jose, California

Andy Dolan
Kyle Haefner
Craig Pratt
Darshak Thakore
CableLabs
Louisville, Colorado

Nick Allot
Ashley Setter
NquiringMinds
Southampton, United Kingdom

May 2024

DRAFT

This publication is available free of charge from
https://www.nccoe.nist.gov/projects/trusted-iot-device-network-layer-onboarding-and-lifecycle-management

DISCLAIMER

Certain commercial entities, equipment, products, or materials may be identified by name or company logo or other insignia in order to acknowledge their participation in this collaboration or to describe an experimental procedure or concept adequately. Such identification is not intended to imply special status or relationship with NIST or recommendation or endorsement by NIST or NCCoE; neither is it intended to imply that the entities, equipment, products, or materials are necessarily the best available for the purpose.

While NIST and the NCCoE address goals of improving management of cybersecurity and privacy risk through outreach and application of standards and best practices, it is the stakeholder's responsibility to fully perform a risk assessment to include the current threat, vulnerabilities, likelihood of a compromise, and the impact should the threat be realized before adopting cybersecurity measures such as this recommendation.

National Institute of Standards and Technology Special Publication 1800-36C, Natl. Inst. Stand. Technol. Spec. Publ. 1800-36C, 55 pages, May 2024, 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: iot-onboarding@nist.gov.

Public comment period: May 31, 2024 through July 30, 2024

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, 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 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.

KEYWORDS

application-layer onboarding; bootstrapping; Internet of Things (IoT); Manufacturer Usage Description (MUD); network-layer onboarding; onboarding; Wi-Fi Easy Connect.

ACKNOWLEDGMENTS

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

Name	Organization
Amogh Guruprasad Deshmukh	Aruba, a Hewlett Packard Enterprise company
Bart Brinkman	Cisco
Eliot Lear	Cisco
Peter Romness	Cisco
Tyler Baker	Foundries.io
George Grey	Foundries.io
David Griego	Foundries.io
Fabien Gremaud	Kudelski IoT
Brecht Wyseur	Kudelski IoT
Faith Ryan	The MITRE Corporation
Toby Ealden	NquiringMinds
John Manslow	NquiringMinds
Antony McCaigue	NquiringMinds
Alexandru Mereacre	NquiringMinds
Loic Cavaille	NXP Semiconductors
Mihai Chelalau	NXP Semiconductors
Julien Delplancke	NXP Semiconductors
Anda-Alexandra Dorneanu	NXP Semiconductors
Todd Nuzum	NXP Semiconductors

Name	Organization
Nicusor Penisoara	NXP Semiconductors
Laurentiu Tudor	NXP Semiconductors
Michael Richardson	Sandelman Software Works
Karen Scarfone	Scarfone Cybersecurity
Steve Clark	SEALSQ, a subsidiary of WISeKey
Pedro Fuentes	SEALSQ, a subsidiary of WISeKey
Gweltas Radenac	SEALSQ, a subsidiary of WISeKey
Kalvin Yang	SEALSQ, a subsidiary of WISeKey
Mike Dow	Silicon Labs
Steve Egerter	Silicon Labs

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 Collaborators		
Aruba, a Hewlett Packard Enterprise company	Foundries.io	Open Connectivity Foundation (OCF)
CableLabs	Kudelski IoT	Sandelman Software Works
Cisco	NquiringMinds	SEALSQ, a subsidiary of WISeKey
	NXP Semiconductors	Silicon Labs

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.

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:

a) 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

b) 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:

1. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
2. 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: iot-onboarding@nist.gov.

Contents

List of Figures

1 Introduction

The following volumes of this guide show information technology (IT) professionals and security engineers how we implemented these example solutions. 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 NIST Cybersecurity Practice Guide demonstrates a standards-based reference design for implementing trusted IoT device network-layer onboarding and lifecycle management and describes various example implementations of this reference design. Each of these implementations, which are known as *builds,* is standards-based and is designed to help provide assurance that networks are not put at risk as new IoT devices are added to them and to help safeguard IoT devices from connecting to unauthorized networks. The reference design described in this practice guide is modular and can be deployed in whole or in part, enabling organizations to incorporate trusted IoT device network-layer onboarding and lifecycle management into their legacy environments according to goals that they have prioritized based on risk, cost, and resources.

NIST is adopting an agile process to publish this content. Each volume is being made available as soon as possible rather than delaying release until all volumes are completed.

This guide contains five volumes:

- NIST Special Publication (SP) 1800-36A: *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 1800-36B: *Approach, Architecture, and Security Characteristics* – what we built and why
- NIST SP 1800-36C: *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)**
- NIST SP 1800-36D: *Functional Demonstrations* – use cases that have been defined to showcase trusted IoT device network-layer onboarding and lifecycle management security capabilities and the results of demonstrating these use cases with each of the example implementations
- NIST SP 1800-36E: *Risk and Compliance Management* – risk analysis and mapping of trusted IoT device network-layer onboarding and lifecycle management security characteristics to cybersecurity standards and recommended practices

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 1800-36A*, which describes the following topics:

- challenges that enterprises face in migrating to the use of trusted IoT device network-layer onboarding
- example solutions built at the NCCoE
- benefits of adopting the example solution

Technology or security program managers who are concerned with how to identify, understand, assess, and mitigate risk will be interested in *NIST SP 1800-36B*, which describes what we did and why.

Also, Section 4 of *NIST SP 1800-36E* will be of particular interest. Section 4, *Mappings*, maps logical components of the general trusted IoT device network-layer onboarding and lifecycle management reference design to security characteristics listed in various cybersecurity standards and recommended practices documents, including *Framework for Improving Critical Infrastructure Cybersecurity* (NIST Cybersecurity Framework) and *Security and Privacy Controls for Information Systems and Organizations* (NIST SP 800-53).

You might share the *Executive Summary, NIST SP 1800-36A*, with your leadership team members to help them understand the importance of using standards-based trusted IoT device network-layer onboarding and lifecycle management implementations.

IT professionals who want to implement similar solutions will find the whole practice guide useful. You can use the how-to portion of the guide, *NIST SP 1800-36C*, to replicate all or parts of the builds created in our lab. The how-to portion of the guide provides specific product installation, configuration, and integration instructions for implementing the example solution. 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. Also, you can use *Functional Demonstrations, NIST SP 1800-36D*, which provides the use cases that have been defined to showcase trusted IoT device network-layer onboarding and lifecycle management security capabilities and the results of demonstrating these use cases with each of the example implementations. Finally, *NIST SP 1800-36E* will be helpful in explaining the security functionality that the components of each build provide.

This guide assumes that IT professionals have experience implementing security products within the enterprise. While we have used a suite of commercial products to address this challenge, this guide does not endorse these particular products. Your organization can adopt this solution 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 trusted IoT device network-layer onboarding and lifecycle management 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 recommended practices.

A NIST Cybersecurity Practice Guide does not describe "the" solution, but example solutions. We seek feedback on the publication's contents and welcome your input. Comments, suggestions, and success stories will improve subsequent versions of this guide. Please contribute your thoughts to iot-onboarding@nist.gov.

1.2 Build Overview

This NIST Cybersecurity Practice Guide addresses the challenge of network-layer onboarding using standards-based protocols to perform trusted network-layer onboarding of an IoT device. Each build demonstrates one or more of these capabilities:

- Trusted Network-Layer Onboarding: providing the device with its unique network credentials over an encrypted channel
- Network Re-Onboarding: performing trusted network-layer onboarding of the device again, after device reset
- Network Segmentation: assigning a device to a particular local network segment to prevent it from communicating with other network components, as determined by enterprise policy
- Trusted Application-Layer Onboarding: providing the device with application-layer credentials over an encrypted channel after completing network-layer onboarding
- Ongoing Device Authorization: continuously monitoring the device on an ongoing basis, providing policy-based assurance and authorization checks on the device throughout its lifecycle
- Device Communications Intent Enforcement: Secure conveyance of device communications intent information, combined with enforcement of it, to ensure that IoT devices are constrained to sending and receiving only those communications that are explicitly required for each device to fulfill its purpose

Five builds that will serve as examples of how to onboard IoT devices using the protocols described in NIST SP 1800-36B, as well as the factory provisioning builds, are being implemented and will be demonstrated as part of this project. The remainder of this practice guide provides step-by-step instructions on how to reproduce all builds.

1.2.1 Reference Architecture Summary

The builds described in this document are instantiations of the trusted network-layer onboarding and lifecycle management logical reference architecture that is described in NIST SP 1800-36B. This architecture is organized according to five high-level processes: Device Manufacture and Factory Provisioning, Device Ownership and Bootstrapping Information Transfer, Trusted Network-Layer Onboarding, Trusted Application-Layer Onboarding, and Continuous Verification. For a full explanation of the architecture, please see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

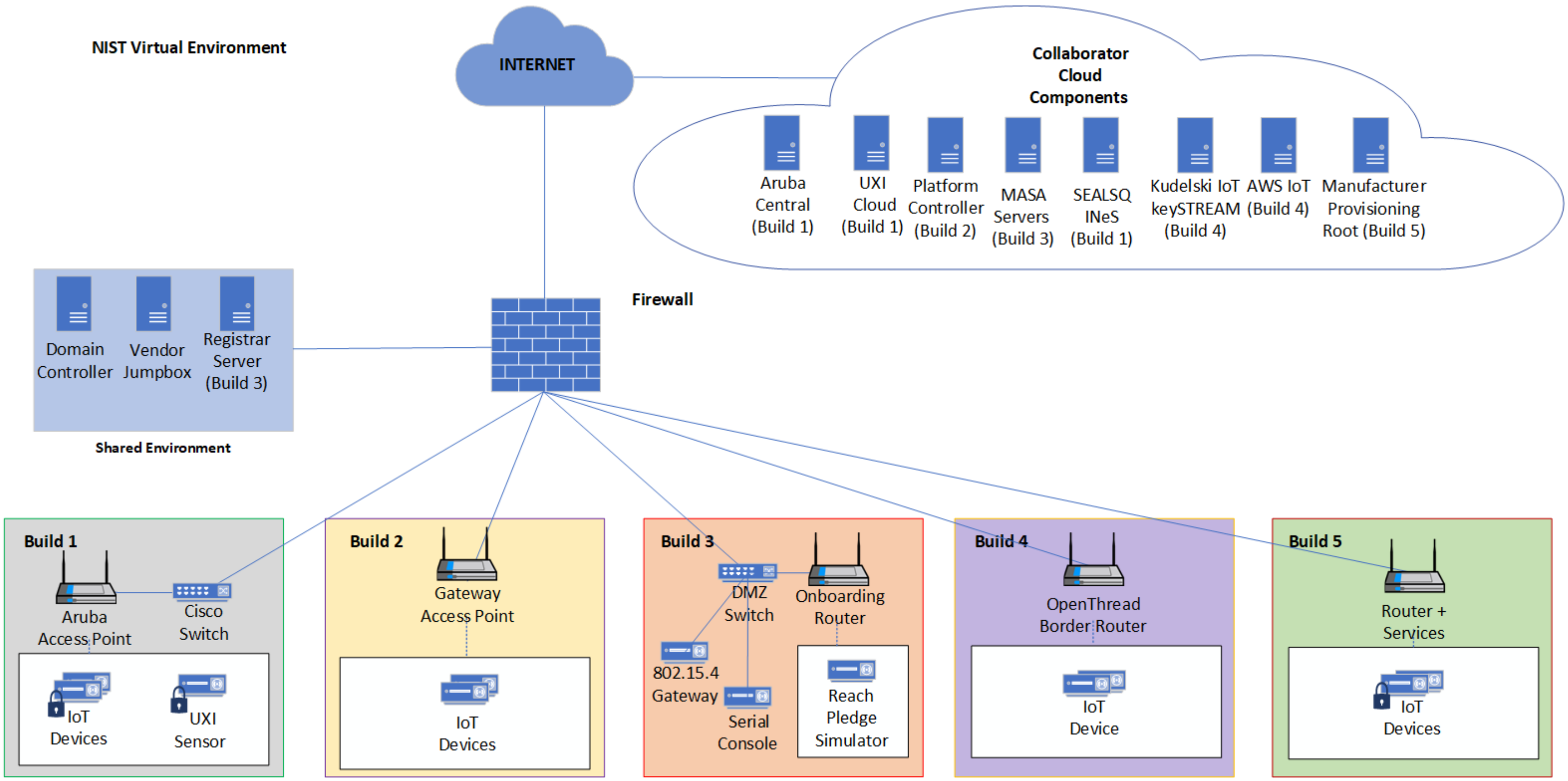
1.2.2 Physical Architecture Summary

Figure 1-1 depicts the high-level physical architecture of the NCCoE IoT Onboarding laboratory environment in which the five trusted IoT device network-layer onboarding project builds and the two factory provisioning builds are being implemented. The NCCoE provides virtual machine (VM) resources and physical infrastructure for the IoT Onboarding lab. As depicted, the NCCoE IoT Onboarding laboratory hosts collaborator hardware and software for the builds. The NCCoE also provides connectivity from the IoT Onboarding lab to the NIST Data Center, which provides connectivity to the internet and public IP spaces (both IPv4 and IPv6). Access to and from the NCCoE network is protected by a firewall.

Access to and from the IoT Onboarding lab is protected by a pfSense firewall, represented by the brick box icon in Figure 1-1. This firewall has both IPv4 and IPv6 (dual stack) configured. The IoT Onboarding lab network infrastructure includes a shared virtual environment that houses a domain controller and a vendor jumpbox. These components are used across builds where applicable. It also contains five independent virtual local area networks (VLANs), each of which houses a different trusted network-layer onboarding build.

The IoT Onboarding laboratory network has access to cloud components and services provided by the collaborators, all of which are available via the internet. These components and services include Aruba Central and the UXI Cloud (Build 1), SEALSQ INeS (Build 1), Platform Controller (Build 2), a MASA server (Build 3), Kudelski IoT keySTREAM application-layer onboarding service and AWS IoT (Build 4), and a Manufacturer Provisioning Root (Build 5).

300 **Figure 1-1 NCCoE IoT Onboarding Laboratory Physical Architecture**



All five network-layer onboarding laboratory environments, as depicted in the diagram, have been installed:

- Build 1 (i.e., the Wi-Fi Easy Connect, Aruba/HPE build) network infrastructure within the NCCoE lab consists of two components: the Aruba Access Point and the Cisco Switch. Build 1 also requires support from Aruba Central for network-layer onboarding and the UXI Cloud for application-layer onboarding. These components are in the cloud and accessed via the internet. The IoT devices that are onboarded using Build 1 include the UXI Sensor and the Raspberry Pi.
- Build 2 (i.e., the Wi-Fi Easy Connect, CableLabs, OCF build) network infrastructure within the NCCoE lab consists of a single component: the Gateway Access Point. Build 2 requires support from the Platform Controller, which also hosts the IoTivity Cloud Service. The IoT devices that are onboarded using Build 2 include three Raspberry Pis.
- Build 3 (i.e., the BRSKI, Sandelman Software Works build) network infrastructure components within the NCCoE lab include a Wi-Fi capable home router (including Join Proxy), a DMZ switch (for management), and an ESP32A Xtensa board acting as a Wi-Fi IoT device, as well as an nRF52840 board acting as an IEEE 802.15.4 device. A management system on a BeagleBone Green serves as a serial console. A registrar server has been deployed as a virtual appliance on the NCCoE private cloud system. Build 3 also requires support from a MASA server which is accessed via the internet. In addition, a Raspberry Pi 3 provides an ethernet/802.15.4 gateway, as well as a test platform.
- Build 4 (i.e., the Thread, Silicon Labs, Kudelski IoT build) network infrastructure components within the NCCoE lab include an Open Thread Border Router, which is implemented using a Raspberry Pi, and a Silicon Labs Gecko Wireless Starter Kit, which acts as an 802.15.4 antenna. Build 4 also requires support from the Kudelski IoT keySTREAM service, which is in the cloud and accessed via the internet. The IoT device that is onboarded in Build 4 is the Silicon Labs Dev Kit (BRD2601A) with an EFR32MG24 System-on-Chip. The application service to which it onboards is AWS IoT.
- Build 5 (i.e., the BRSKI over Wi-Fi, NquiringMinds build) includes 2 Raspberry Pi 4Bs running a Linux operating system. One Raspberry Pi acts as the pledge (or IoT Device) with an Infineon TPM connected. The other acts as the router, registrar and MASA all in one device. This build uses the open source TrustNetZ distribution, from which the entire build can be replicated easily. The TrustNetZ distribution includes source code for the IoT device, the router, the access point, the network onboarding component, the policy engine, the manufacturer services, the registrar and a demo application server. TrustNetZ makes use of NquiringMinds tdx Volt to issue and validate verifiable credentials.
- The BRSKI factory provisioning build is deployed in the Build 5 environment. The IoT device in this build is a Raspberry Pi equipped with an Infineon Optiga SLB 9670 TPM 2.0, which gets provisioned with birth credentials (i.e., a public/private key pair and an IDevID). The BRSKI factory provisioning build also uses an external certificate authority hosted on the premises of NquiringMinds to provide the device certificate signing service.
- The Wi-Fi Easy Connect factory provisioning build is deployed in the Build 1 environment. Its IoT devices are Raspberry Pis equipped with a SEALSQ VaultIC Secure Element, which gets provisioned with a DPP URI. The Secure Element can also be provisioned with an IDevID certificate signed by the SEALSQ INeS certification authority, which is independent of the DPP URI. Code for performing the factory provisioning is stored on an SD card.

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 (Wi-Fi Easy Connect, Aruba/HPE)

This section of the practice guide contains detailed instructions for installing and configuring all the products used to build an instance of the example solution. For additional details on Build 1's logical and physical architectures, see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

The network-layer onboarding component of Build 1 utilizes Wi-Fi Easy Connect, also known as the Device Provisioning Protocol (DPP). The Wi-Fi Easy Connect standard is maintained by the Wi-Fi Alliance [1]. The term "DPP" is used when referring to the network-layer onboarding protocol, and "Wi-Fi Easy Connect" is used when referring to the overall implementation of the network onboarding process.

2.1 Aruba Central/Hewlett Packard Enterprise (HPE) Cloud

This build utilized Aruba Central as a cloud management service that provided management and support for the Aruba Wireless Access Point (AP) and provided authorization and DPP onboarding capabilities for the wireless network. A cloud-based application programming interface (API) endpoint provided the ability to import the DPP Uniform Resource Identifiers (URIs) in the manner of a Supply Chain Integration Service. Due to this capability and Build 1's support for Wi-Fi Easy Connect, Build 1's infrastructure fully supported interoperable network-layer onboarding with Build 2's Reference Clients ("IoT devices") provided by CableLabs.

2.2 Aruba Wireless Access Point

Use of DPP is implicitly dependent on the Aruba Central cloud service. Aruba Central provides a cloud Infrastructure as a Service (IaaS) enabled architecture that includes initial support for DPP in Central 2.5.6/ArubaOS (AOS) 10.4.0. Central and AOS support multiple deployment formats:

1. As AP only, referred to as an *underlay deployment,* where traffic is bridged locally from the APs.

2. An *overlay deployment,* where all data is securely tunneled to an on-prem gateway where advanced services can route, inspect, and analyze the data before it's either bridged locally or routed to its next hop.

3. A *mixed-mode deployment,* which is a combination of the two where a returned 'role/label' is used to determine how the data is processed and forwarded.

At the time of this publication, a user can leverage any 3xx, 5xx, or 6xx APs to support a DPP deployment, with a view that all future series APs will implicitly include support. For an existing or new user there is a prerequisite of the creation of a Service Set Identifier (SSID). Note that DPP today is not supported under Wi-Fi Protected Access 3 (WPA3); this is a roadmap item with no published timeline.

Assuming there is an existing SSID or a new one is created based upon the above security restrictions, the next step is to enable DPP (as detailed below in Section 2.2.1) such that the SSID can support multiple authentication and key managements (AKMs) on a Basic Service Set (BSS). If the chosen security type is DPP, only a single AKM will exist for that BSS.

A standards-compliant 802.3at port is the easiest method for providing the AP with power. An external power supply can also be used.

Within this document, we do not cover the specifics of radio frequency (RF) design and placement of APs. Guidance and assistance is available within the Aruba community site, https://community.arubanetworks.com or the Aruba Support Portal, https://asp.arubanetworks.com. Additionally, we do not cover onboarding and licensing of Aruba Central hardware. Documentation can be found here: https://www.arubanetworks.com/techdocs/ArubaDocPortal/content/docportal.htm.

2.2.1 Wi-Fi Network Setup and Configuration

The following instructions detail the initial setup and configuration of the Wi-Fi network upon powering on and connecting the AP to an existing network.

1. Navigate to the Aruba Central cloud management interface.

2. On the sidebar, navigate under **Global** and choose the AP-Group you want to configure/modify. (This assumes you have already grouped your APs by location/functions.)

3. Under **Devices,** click **Config** in the top right side.

4. You will now be in the Access Points tab and WLANs tab. Do one of the following:

    a. If creating a new SSID, click on **+ Add SSID.** After entering the Name (SSID) in Step 1 and configuring options as necessary in Step 2, when you get to Step 3 (Security), it will default on the slide-bar to the Personal Security Level; the alternative is the Enterprise Security Level.

        i. If you choose the **Personal Security Level,** under **Key-Management** ensure you select either **DPP** or **WPA2-Personal.** If you choose **WPA2-Personal,** expand the **Advanced Settings** section and enable the toggle button for DPP so that the SSID can broadcast the AKM. Note that this option is not available if choosing DPP for Key-Management.

   ii. If you choose the **Enterprise Security Level,** only WPA2-Enterprise Key-Management currently supports DPP. Expand the **Advanced Settings** section and enable the toggle button for **DPP** so that the SSID can broadcast the AKM.

b. If you plan to enable DPP on a previously created SSID:

   i. Ensure you are running version 10.4+ on your devices. You also need an SSID that is configured for WPA2-Personal or WPA2-Enterprise.

   ii. When ready, float your cursor over the previously created SSID name you wish to configure and click on the edit icon.

   iii. Edit the SSID, click on **Security**, and expand the **Advanced Settings** section and enable the toggle button for **DPP.**

   iv. Click **Save Settings.**

For SSIDs that have been modified to add DPP AKM, it's also necessary to enable DPP within the radio profile.

1. Under the **Access Point** Tab, click **Radios.**
2. It's expected you'll see a **default** radio-profile. If a custom one has been created, you'll need to review your configuration before proceeding.
3. Assuming a **default** radio-profile, click on the **Edit** icon, expand **Show advanced settings,** and scroll down to **DPP Provisioning.** You can selectively enable this for 2.4 GHz or 5.0 GHz. Support for DPP on 6.0 GHz is a roadmap item at this time and is not yet available.

2.2.2 Wi-Fi Easy Connect Configuration

Configuration of the Access Point occurred through the Aruba Central cloud management interface. Standard configurations were used to stand up the Build 1 wireless network. The instructions for enabling DPP capabilities for the overall wireless network are listed below:

1. Navigate to the Aruba Central cloud management interface.
2. On the sidebar, navigate to **Security > Authentication and Policy > Config.**
3. In the **Client Access Policy** section, click **Edit.**
4. Under the **Wi-Fi Easy Connect™ Service** heading, ensure that the name of your wireless network is selected.
5. Click **Save.**

2.3 Cisco Catalyst 3850-S Switch

This build utilized a Cisco Catalyst 3850-S switch. This switch utilized a minimal configuration with two separate VLANs to allow for IoT device network segmentation and access control. The switch also provided Power-over-Ethernet support for the Aruba Wireless AP.

2.3.1 Configuration

The switch was configured with two VLANs, and a trunk port dedicated to the Aruba Wireless AP. You can find the relevant portions of the Cisco iOS configuration below:

```
interface Vlan1
 no ip address
interface Vlan2
 no ip address
interface GigabitEthernet1/0/1
 switchport mode trunk
interface GigabitEthernet1/0/2
 switchport mode access
 switchport access vlan 1
interface GigabitEthernet1/0/3
 switchport mode access
 switchport access vlan 2
```

2.4 Aruba User Experience Insight (UXI) Sensor

This build utilized an Aruba UXI Sensor as a Wi-Fi Easy Connect-capable IoT device. Models G6 and G6C support Wi-Fi Easy Connect, and all available G6 and G6C models support Wi-Fi Easy Connect within their software image. This sensor successfully utilized the network-layer onboarding mechanism provided by the wireless network and completed onboarding to the application-layer UXI cloud service. The network-layer onboarding process is automatically initiated by the device on boot.

2.4.1 Configuration

All of Aruba's available G6 and G6C UXI sensors support the ability to complete network-layer and application-layer onboarding. No specific configuration of the physical sensor is required. As part of the supply-chain process, the cryptographic public key for your sensor(s) will be available within the cloud tenant. This public/private keypair for each device is created as part of the manufacturing process. The public key effectively identifiers the sensor to the network and as part of the Wi-Fi Easy Connect/DPP onboarding process. This allows unprovisioned devices straight from the factory to be onboarded and subsequently connect to the UXI sensor cloud to obtain their network-layer configuration. An administrator will have to define the 'tasks' the UXI sensor is going to perform such as monitoring SSIDs, performing reachability tests to on-prem or cloud services, and making the results of these tests available within the UXI user/administrator portal.

2.5 Raspberry Pi

In this build, the Raspberry Pi 3B+ acts as a DPP enrollee. In setting up the device for this build, a DPP-capable wireless adapter, the Alfa AWUS036NHA network dongle, was connected to enable the Pi to send and receive DPP frames. Once fully configured, the Pi can onboard with the Aruba AP.

2.5.1 Configuration

The following steps were completed for the Raspberry Pi to complete DPP onboarding:

1. Set the management IP for the Raspberry Pi to an IP address in the Build 1 network. To do this, add the following lines to the file *dhcpcd.conf* located at */etc/dhcpcd.conf*. For this build, the IP address was set to 192.168.10.3.

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.10.3/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.10.1
static domain_name_servers=192.168.10.1 8.8.8.8
```

2. Install Linux Libraries using the apt package manager. The following packages were installed:
   a. autotools-dev
   b. automake
   c. libcurl4-openssl-dev
   d. libnl-genl-3-dev
   e. libavahi-client-dev
   f. libavahi-core-dev
   g. aircrack-ng
   h. openssl-1.1.1q
3. Install the DPP utilities. These utilities were installed from the GitHub repository https://github.com/HewlettPackard/dpp using the following command:

```
git clone https://github.com/HewlettPackard/dpp
```

2.5.2 DPP Onboarding

This section describes the steps for using the Raspberry Pi as a DPP enrollee. The Pi uses a DPP utility to send out chirps to make its presence known to available DPP configurators. Once the Pi is discovered, the DPP configurator (Aruba Wireless AP) initiates the DPP authentication protocol. During this phase, DPP *connectors* are created to onboard the device to the network. As soon as the Pi is fully authenticated, it is fully enrolled and can begin normal network communication.

1. Navigate to the DPP utilities directory which was installed during setup:

```
cd dpp/linux
```

2. From the DPP utilities directory, run the following command to initiate a DPP connection:

```
sudo ./sss -I wlan1 -r -e sta -k respp256.pem -B respbkeys.txt -a -t -d 255
```

```
build1@Build1Pi:~/dpp/linux $ sudo ./sss -I wlan1 -r -e sta -k respp256.pem -B respbkeys.txt -a -t -d 255
adding interface wlan1...
wlan1 is NOT the loopback!

getting the interface!
got phy info!!!
interface MAC address is 00:c0:ca:98:42:37
wiphy is 1
wlan1 is interface 4 from ioctl
wlan1 is interface 4 from if_nametoindex()
max ROC is 5000
got driver capabilities, off chan is ok, max_roc is 5000

ask for GAS request frames

ask for GAS response frames

ask for GAS comeback request frames

ask for GAS comeback response frames

ask for DPP action frames
socket 4 is for nl_sock_in
role: enrollee
interfaces and MAC addresses:
        wlan1: 00:c0:ca:98:42:37
chirping, so scan for APs
scanning for all SSIDs
scan finished.
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
didn't find the DPP Configurator connectivity IE on
FOUND THE DPP CONFIGURATOR CONNECTIVITY IE on Build1-IoTOnboarding, on frequency 2462, channel 11
```

3. Once the enrollee has found a DPP configurator, the DPP authentication protocol is initiated.

```
------- Start of DPP Authentication Protocol ---------
chirp list:
        2437
        2412
        2462
start chirping...
error...-95: Unspecific failure
changing frequency to 2437
sending 68 byte frame on 2437
chirp on 2437...
error...-95: Unspecific failure
changing frequency to 2412
sending 68 byte frame on 2412
chirp on 2412...
error...-95: Unspecific failure
changing frequency to 2462
sending 68 byte frame on 2462
chirp on 2462...
processing 222 byte incoming management frame
enter process_dpp_auth_frame() for peer 1
        peer 1 is in state DPP bootstrapped
Got a DPP Auth Frame! In state DPP bootstrapped
type Responder Bootstrap Hash, length 32, value:
05d54478 eaa59dfa 768d8148 f119f729 060c8d3b b9e917dc 4b34d654 32f403cb

type Initiator Bootstrap Hash, length 32, value:
2795ec93 1b5b17c9 e0e5e5ad b2ce787d 413ab0c2 bb29cfbf 554668fe a090eeea

type Initiator Protocol Key, length 64, value:
bbb37f18 0839880d 7d5bb455 c6702cde fe51d0ee 2c93b895 0edb368d 23d9eca1
d8fc9568 c7af6542 e97aeeb4 bbae7885 05745f8d 82cac4c5 376cc6fb 30d956af

type Protocol Version, length 1, value:
02

type Wrapped Data, length 41, value:
62ceb78b 1b27d2d0 726b9f12 918736a3 ba0d8c68 00ab1509 9e2ebbc5 e61250fe
b90fc9e3 0e97cd5b b6

responder received DPP Auth Request
peer sent a version of 2
Pi'
x:
bbb37f18 0839880d 7d5bb455 c6702cde fe51d0ee 2c93b895 0edb368d 23d9eca1

y:
d8fc9568 c7af6542 e97aeeb4 bbae7885 05745f8d 82cac4c5 376cc6fb 30d956af

k1:
8de1c000 01b44e44 dbaf5bd5 273f4621 bb33bd6f f48e1dc1 3db71ba2 8852d293

initiator's nonce:
378708d9 2985f2a6 239e7ffa 0ee1649a

initiator role: configurator
my role: enrollee
```

2.6 Certificate Authority

The function of the certificate authority (CA) in this build is to issue network credentials for use in the network-layer onboarding process.

2.6.1 Private Certificate Authority

A private CA was provided as a part of the DPP demonstration utilities in the HPE GitHub repository. For demonstration purposes, the Raspberry Pi is used as the configurator and the enrollee.

2.6.1.1 Installation and Configuration

The following instructions detail the initial setup and configuration of the private CA using the DPP demonstration utilities and certificates located at https://github.com/HewlettPackard/dpp.

1. Navigate to the DPP utilities directory on the Raspberry Pi: *~dpp/linux*

    ```
    cd dpp/linux/
    ```

2. The README in the GitHub repository (https://github.com/HewlettPackard/dpp/blob/master/README) references a text file called *configakm* which contains information about the network policies for a configurator to provision on an enrollee. The format is: <akm> <EAP server> <ssid>. Current AKMs that are supported are DPP, dot1x, sae, and psk. For this build, DPP is used. For DPP, an Extensible Authentication Protocol (EAP) server is not used.

3. Configure the file *configakm* located in *~/dpp/linux/*. This file instructs the configurator on how to deploy a DPP connector (network credential) from the configurator to the enrollee. As shown below, the *configakm* file is filled with the following fields:
    dpp unused Build1-IoTOnboarding.

4. The file *csrattrs.conf* contains attributes to construct an Abstract Syntax Notation One (ASN.1) string. This string allows the configurator to tell the enrollee how to generate a certificate signing request (CSR). The following fields were used for this demonstration:

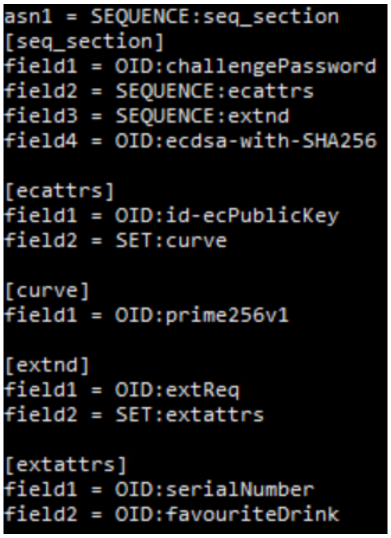
    ```
    asn1 = SEQUENCE: seq_section
    [seq_section]
    field1 = OID:challengePassword
    field2 = SEQUENCE:ecattrs
    field3 = SEQUENCE:extnd
    field4 = OID:ecdsa-with-SHA256

    [ecattrs]
    field1 = OID:id-ecPublicKey
    field2 = SET:curve

    [curve]
    field1 = OID:prime256v1
    ```

```
[extnd]
field1 = OID:extReq
field2 = SET:extattrs

[extattrs]
field1 = OID:serialNumber
field2 = OID:favouriteDrink
```

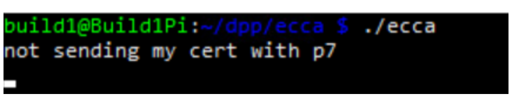


2.6.1.2 Operation and Demonstration

Once setup and configuration have been completed, the following steps can be used to demonstrate utilizing the private CA to issue credentials to a requesting device.

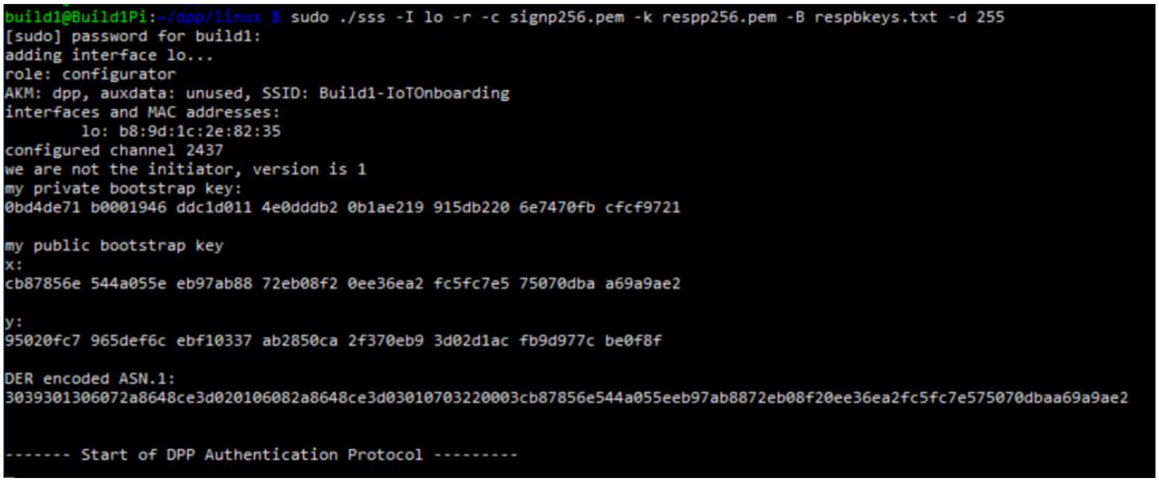
1. Open three terminals on the Raspberry Pi: one to start the certificate program, one to show the configurator's point of view, and one to show the enrollee's point of view.
2. The demonstration uses an OpenSSL certificate. To run the program from the first terminal, navigate to the following directory: *~/dpp/ecca/,* and run the command:

```
./ecca.
```

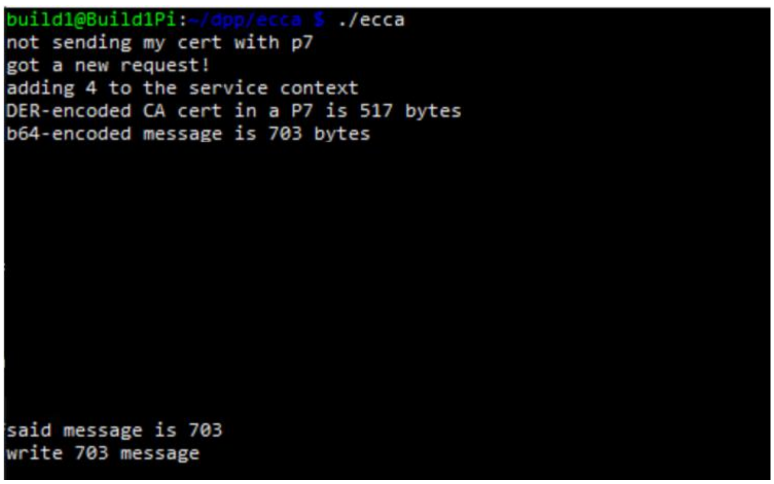


3. On the second terminal, start the configurator using the following command:

```
sudo ./sss -I lo -r -c signp256.pem -k respp256.pem -B resppbkeys.txt -d 255
```



As shown in the terminal where the ecca program is running, the configurator contacts the CA and asks for the certificate.



4. On the third terminal, start the enrollee using the following command:

```
sudo ./sss -I lo -r -e sta -k initp256.pem -B initbkeys.txt -t -a -q -d 255
```

From the enrollee's perspective, it will send chirps on different channels until it finds the configurator. Once found, it sends its certificate to the CA for signing. The snippet below is of the enrollee generating the CSR.

```
authenticated initiator!
start the configuration protocol....
exit process_dpp_auth_frame() for peer 1
        peer 1 is in state DPP authenticated
beginning DPP Config protocol
sending a GAS_INITIAL_REQUEST dpp config frame
processing 198 byte incoming management frame
got a GAS_INITIAL_RESPONSE...
response len is 155, comeback delay is 0
got a DPP config response!
Configurator said we need a CSR to continue...
CSR Attributes:
4d457747 43537147 53496233 4451454a 427a4156 42676371 686b6a4f 50514942
4d516f47 43437147 534d3439 41774548 4d423447 43537147 53496233 4451454a
0a446a45 5242674e 56424155 4743676d 534a6f6d 54386978 6b415155 47434371
47534d34 3942414d 430a

adding 88 byte challengePassword
an object, not an attribute
a nid for challengePassword
CSR Attr parse: got a SET OF attributes... nid for ecPublicKey
 an elliptic curve, nid = 415
CSR Attr parse: got a SET OF attributes... an extension request:
        for serial number
        for favorite drink
an object, not an attribute
a nid for ecdsa with sha256
using bootstrapping key for CSR...
CSR is 537 chars:
```

5. In the ecca terminal, the certificate from the enrollee is shown

```
Write out database with 1 new entries
Data Base Updated
DER-encoded P7 is 681 bytes
b64-encoded message is 923 bytes











said message is 923
write 923 message
```

2.6.2 SEALSQ INeS

The SEALSQ INeS Certificate Management System provides CA and certificate management capabilities for Build 1. Implementation of this system provides Build 1 with a trusted, public CA to support issuing network credentials.

2.6.2.1 Setup and Configuration

To support this build, a custom software agent was deployed on a Raspberry Pi in the Build 1 network. This agent interacted with the cloud-based CA in SEALSQ INeS via API to sign network credentials. Network-level onboarding of IoT devices was completed via DPP, with network credentials being successfully requested from and issued by SEALSQ INeS.

Additional information on interacting with the SEALSQ INeS API can be found at https://inesdev.certifyiddemo.com/. Access can be requested directly from SEALSQ via their contact form: https://www.sealsq.com/contact.

2.7 UXI Cloud

The UXI Cloud is a web-based application that serves as a monitoring hub for the UXI sensor. It provides visibility into the data captured by the performance monitoring that the UXI sensor conducts. For the purposes of this build, the dashboard was used to demonstrate application-layer onboarding, which occurs once the UXI sensor has completed network-layer onboarding. Once application-layer onboarding was completed and the application configuration had been applied to the device, our demonstration concluded.

2.8 Wi-Fi Easy Connect Factory Provisioning Build

This Factory Provisioning Build included many of the components listed above, including Aruba Central, SEALSQ INeS, the Aruba Access Point, and Raspberry Pi IoT devices. A SEALSQ VaultIC Secure Element was also included in the build and provided secure generation and storage of the key material and certificates provisioned to the device.

2.8.1 SEALSQ VaultIC Secure Element

The SEALSQ VaultIC Secure Element was connected to a Raspberry Pi via the built-in GPIO pins present on the Pi. SEALSQ provided demonstration code that generates a public/private keypair within the secure element, creates a Certificate Signing Request, and uses that CSR to obtain an IDevID certificate from SEALSQ INeS. This code supports the Raspberry Pi OS Bullseye. The demonstration code can be found at the official GitHub repository.

HPE also provided a custom DPP-based implementation of the SEALSQ code, which generates supporting material within the secure element, and then generates a DPP URI. This DPP URI is available in a string format, PNG (QR Code), and ASCII (QR Code). The DPP URI can then be used for network onboarding, as described in the rest of the Build 1 section. This code is included in the demonstration code located at the repository linked above.

2.8.1.1 Installation and Configuration

Full instructions for installation and configuration can be found in the INSTALL.txt file from the SEALSQ demonstration code mentioned above. A general set of steps for preparing to run the demonstration code is included below.

1. Install prerequisites on Raspberry Pi
    a. cmake
    b. git
    c. gcc
2. On the Raspberry Pi, run the `sudo raspi-update` command to update drivers

3. Before plugging VaultIC Secure Element into the Raspberry Pi connector, configure the jumpers:
    a. Set _VCC_ jumper
        i. CTRL = VaultIC power controlled by GPIO25 (default)
        ii. 3V3 = VaultIC power always on
    b. Set J1&J2 to select I2C or SPI
        i. If using SPI, set J1 to SS and J2 to SEL (default)
        ii. If using I2C, set J1 to SCL and J2 to SDA
4. Using the `raspi-config` command, enable the SPI or I2C interface on the Raspberry Pi
5. Run `git clone https://github.com/sclark-wisekey/NCCoE.factory.pub` to pull down the demonstration code.

2.8.1.2 Running the demonstration code

1. Navigate to the folder containing the demonstration code. Inside that folder, navigate to the VaultIC/demos folder.
2. Edit the file config.cfg and change the value of VAULTIC_COMM to match with the jumpers configured during setup.
3. The demonstrations are available with wolfSSL stacks and organized in dedicated folders. The README.TXT file in each demonstration subfolder explains how to run the demonstrations.

3 Build 2 (Wi-Fi Easy Connect, CableLabs, OCF)

This section of the practice guide contains detailed instructions for installing and configuring all of the products used to build an instance of the example solution. For additional details on Build 2's logical and physical architectures, see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

The network-layer onboarding component of Build 2 utilizes Wi-Fi Easy Connect, also known as the Device Provisioning Protocol (DPP). The Wi-Fi Easy Connect standard is maintained by the Wi-Fi Alliance [1]. The term "DPP" is used when referring to the network-layer onboarding protocol, and "Wi-Fi Easy Connect" is used when referring to the overall implementation of the network onboarding process.

3.1 CableLabs Platform Controller

The CableLabs Platform Controller provides an architecture and reference implementation of a cloud-based service that provides management capability for service deployment groups, access points with the deployment groups, registration and lifecycle of user services, and the secure onboarding and lifecycle management of users' Wi-Fi devices. The controller also exposes APIs for integration with third-party systems for the purpose of integrating various business flows (e.g., integration with manufacturing process for device management).

The Platform Controller would typically be hosted by the network operator or a third-party service provider. It can be accessed via web interface. Additional information for this deployment can be accessed at the official CableLabs repository.

3.1.1 Operation and Demonstration

Once configuration of the Platform Controller, Gateway, and Reference Client has been completed, full operation can commence. Instructions for this are located at the official CableLabs repository.

3.2 CableLabs Custom Connectivity Gateway

In this deployment, the gateway software is running on a Raspberry Pi 3B+, which acts as a router, firewall, wireless access point, Open Connectivity Foundation (OCF) Diplomat, and OCF Onboarding Tool. The gateway is also connected to the CableLabs Platform Controller, which manages much of the configuration and functions of the gateway. Due to Build 2's infrastructure and support of Wi-Fi Easy Connect, Build 2 fully supported interoperable network-layer onboarding with Build 1's IoT devices.

3.2.1 Installation and Configuration

Hardware requirements, pre-installation steps, installation steps, and configuration instructions for the gateway can be found at the official CableLabs repository.

3.2.2 Integration with CableLabs Platform Controller

Once initial configuration has occurred, the gateway can be integrated with the CableLabs Platform Controller. Instructions can be found at the official CableLabs repository.

3.2.3 Operation and Demonstration

Once configuration of the Platform Controller, Gateway, and Reference Client has been completed, full operation can commence. Instructions for this are located at the official CableLabs repository.

3.3 Reference Clients/IoT Devices

Three reference clients were deployed in this build, each on a Raspberry Pi 3B+. They were each configured to emulate either a smart light switch or a smart lamp. The software deployed also included the capability to perform network-layer onboarding via Wi-Fi Easy Connect (or DPP) and application-layer onboarding using the OCF onboarding method. These reference clients were fully interoperable with network-layer onboarding to Build 1.

3.3.1 Installation and Configuration

Hardware requirements, pre-installation, installation, and configuration steps for the reference clients are detailed in the official CableLabs repository.

3.3.2 Operation and Demonstration

Once configuration of the Platform Controller, Gateway, and Reference Client has been completed, full operation can commence. Instructions for this are located at the official CableLabs repository.

For interoperability with Build 1, the IoT device's DPP URI was provided to Aruba Central, which allowed Build 1 to successfully complete network-layer onboarding with the Build 2 IoT devices.

4 Build 3 (BRSKI, Sandelman Software Works)

This section of the practice guide contains detailed instructions for installing and configuring all of the products used to build an instance of the example solution. For additional details on Build 3's logical and physical architectures, see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

The network-layer onboarding component of Build 3 utilizes the Bootstrapping Remote Secure Infrastructure (BRSKI) protocol. Build 3 is representative of a typical home or small office network.

4.1 Onboarding Router/Join Proxy

The onboarding router quarantines the IoT device attempting to join the network until the BRSKI onboarding process is complete. The router in this build is a Turris MOX device, which is based on the Linux OpenWrt version 4 operating system (OS). The Raspberry Pi 3 contains software to function as the Join Proxy for pledges to the network. If another brand of device is used, a different source of compiled Join Proxy might be required.

4.1.1 Setup and Configuration

The router needs to be IPv6 enabled. In the current implementation, the join package operates on an unencrypted network.

4.2 Minerva Join Registrar Coordinator

The purpose of the Join Registrar is to determine whether a new device is allowed to join the network. The Join Registrar is located on a virtual machine running Devuan Linux 4 within the network.

4.2.1 Setup and Configuration

The Minerva Fountain Join Registrar/Coordinator is available as a Docker container and as a VM in OVA format at the Minerva fountain page. Further setup and configuration instructions are available on the Sandelman website on the configuration page.

For the Build 3 demonstration, the VM deployment was installed onto a VMware vSphere system.

A freshly booted VM image will do the following on its own:

- Configure a database
- Configure a local certificate authority (fountain:s0_setup_jrc)
- Configure certificates for the database connection
- Configure certificates for the Registrar https interface
- Configure certificates for use with the Bucardo database replication system
- Configure certificates for LDevID certification authority (fountain:s2_create_registrar)

- Start the JRC

The root user is permitted to log in on the console ("tty0") using the password "root" but is immediately forced to set a new password.

The new registrar will announce itself with the name minerva-fountain.local in mDNS.

The logs for this are put into */var/log/configure-fountain-12345.log* (where 12345 is a new number based upon the PID of the script).

4.3 Reach Pledge Simulator

The Reach Pledge Simulator acts as an IoT device in Build 3. The pledge is acting as an IoT device joining the network and is hosted on a Raspberry Pi 3. More information is available on the Sandelman website on the Reach page.

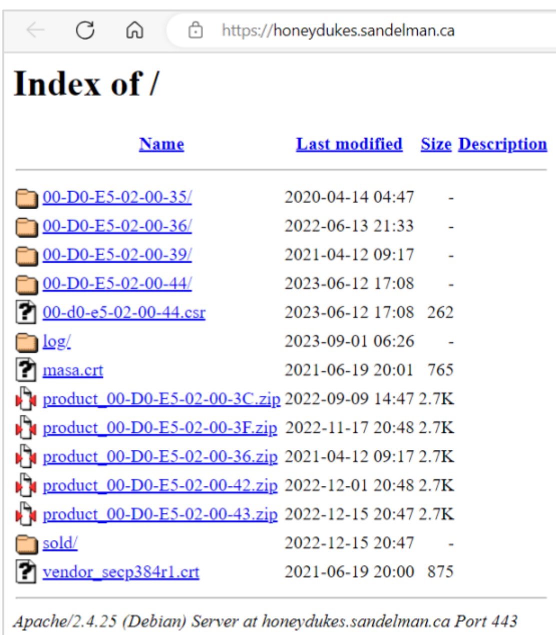
4.3.1 Setup and Configuration

While the functionality of this device is to act as an IoT device, it runs on the same software as the Join Registrar Coordinator. This software is available in both VM and Docker container format. Please see Section 4.2.1 for installation instructions.

When setting up the Reach Pledge Simulator, the address of the Join Registrar Coordinator is automatically determined by the pledge.

Currently, the Reach Pledge Simulator obtains its IDevID using the following steps:

1. View the available packages by visiting the Sandelman website.


https://honeydukes.sandelman.ca

Index of /

Name	Last modified	Size	Description
00-D0-E5-02-00-35/	2020-04-14 04:47	-	
00-D0-E5-02-00-36/	2022-06-13 21:33	-	
00-D0-E5-02-00-39/	2021-04-12 09:17	-	
00-D0-E5-02-00-44/	2023-06-12 17:08	-	
00-d0-e5-02-00-44.csr	2023-06-12 17:08	262	
log/	2023-09-01 06:26	-	
masa.crt	2021-06-19 20:01	765	
product_00-D0-E5-02-00-3C.zip	2022-09-09 14:47	2.7K	
product_00-D0-E5-02-00-3F.zip	2022-11-17 20:48	2.7K	
product_00-D0-E5-02-00-36.zip	2021-04-12 09:17	2.7K	
product_00-D0-E5-02-00-42.zip	2022-12-01 20:48	2.7K	
product_00-D0-E5-02-00-43.zip	2022-12-15 20:47	2.7K	
sold/	2022-12-15 20:47	-	
vendor_secp384r1.crt	2021-06-19 20:00	875	

Apache/2.4.25 (Debian) Server at honeydukes.sandelman.ca Port 443

2. Open a terminal on the Raspberry Pi device and navigate to the Reach directory by entering:

```
cd reach
```

```
nccoe@satine:~$ ls
bin  minerva  reach
nccoe@satine:~$ cd reach
nccoe@satine:~/reach$
```

3. Enter the following command while substituting the URL for one of the available zip files containing the IDevID of choice on the Sandelman website.

    ```
    wget https://honeydukes.sandelman.ca/product_00-D0-E5-02-00-42.zip
    ```

    ```
    nccoe@satine:~/reach$ wget https://honeydukes.sandelman.ca/product_00-D0-E5-02-00-42.zip
    --2023-09-01 15:49:54--  https://honeydukes.sandelman.ca/product_00-D0-E5-02-00-42.zip
    Resolving honeydukes.sandelman.ca (honeydukes.sandelman.ca)... 2a01:7e00:e000:2bb::3d:b021, 176.58.120.209
    Connecting to honeydukes.sandelman.ca (honeydukes.sandelman.ca)|2a01:7e00:e000:2bb::3d:b021|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2722 (2.7K) [application/zip]
    Saving to: ‘product_00-D0-E5-02-00-42.zip’

    product_00-D0-E5-02-00-42.zip 100%[=============================================>]   2.66K  --.-KB/s    in 0.001s

    2023-09-01 15:49:57 (3.27 MB/s) - ‘product_00-D0-E5-02-00-42.zip’ saved [2722/2722]
    ```

4. Unzip the file by entering the following command, substituting the name of your zip file (the IDevID is the *device.crt* file):

    ```
    unzip product_00-D0-E5-02-00-42.zip
    ```

    ```
    nccoe@satine:~/reach$ unzip product_00-D0-E5-02-00-42.zip
    Archive:  product_00-D0-E5-02-00-42.zip
       creating: 00-D0-E5-02-00-42/
      inflating: 00-D0-E5-02-00-42/device.crt
      inflating: 00-D0-E5-02-00-42/masa.crt
      inflating: 00-D0-E5-02-00-42/vendor.crt
      inflating: 00-D0-E5-02-00-42/key.pem
    ```

Typically, this would be accomplished through a provisioning process involving a Certificate Authority, as demonstrated in the Factory Provisioning builds.

4.4 Serial Console Server

The serial console server does not participate in the onboarding process but provides direct console access to the IoT devices. The serial console server has been attached to a multi-port USB hub and USB connectors and/or USB2TTL adapters connected to each device. The ESP32 and the nRF52840 are both connected to the serial console and receive power from the USB hub. Power to the console and IoT devices is also provided via the USB hub. A BeagleBone Green device was used as the serial console, using the "screen" program as the telecom device.

4.5 Minerva Highway MASA Server

In the current implementation of the build, the MASA server provides the Reach Pledge Simulator with an IDevID Certificate and a public/private keypair for demonstration purposes. Typically, this would be accomplished through a factory provisioning process involving a Certificate Authority, as demonstrated in the Factory Provisioning builds.

4.5.1 Setup and Configuration

Installation of the Minerva Highway MASA is described at the Highway configuration page. Additional configuration details are available at the Highway development page.

Availability of VMs and containers is described at the following Minerva page.

5 Build 4 (Thread, Silicon Labs, Kudelski IoT)

This section of the practice guide contains detailed instructions for installing and configuring all of the products used to build an instance of the example solution. For additional details on Build 4's logical and physical architectures, see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

This build utilizes the Thread protocol and performs application-layer onboarding using the Kudelski keySTREAM service to provision a device to the AWS IoT Core.

5.1 Open Thread Border Router

The Open Thread Border Router forms the Thread network and acts as the router on this build. The Open Thread Border Router is run as software on a Raspberry Pi 3B. The Silicon Labs Gecko Wireless Devkit is attached to the Raspberry Pi via USB and acts as the 802.15.4 antenna for this build.

5.1.1 Installation and Configuration

On the Raspberry Pi, run the following commands from a terminal to install and configure the Open Thread Border Router software:

```
git clone https://github.com/openthread/ot-br-posix
sudo NAT64=1 DNS64=1 WEB_GUI=1 ./script/bootstrap
sudo NAT64=1 DNS64=1 WEB_GUI=1 ./script/setup
```

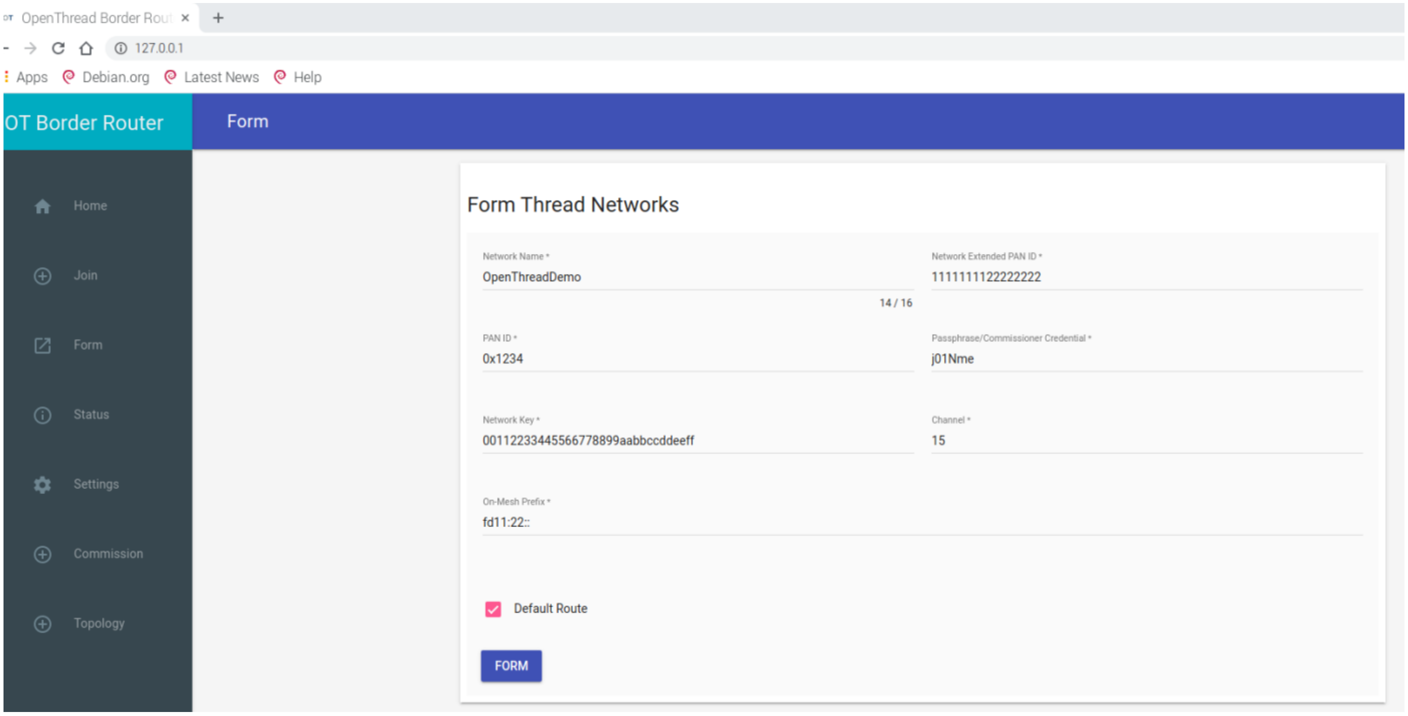
5.1.2 Operation and Demonstration

Once initial configuration has occurred, the OpenThread Border Router should be functional and operated through the web GUI.

1. To open the OpenThread Border Router GUI enter the following IP in a web browser:

   ```
   127.0.0.1
   ```

2. In the **Form** tab, enter the details for the Thread network being formed. For demonstration purposes we only updated the credentials field.



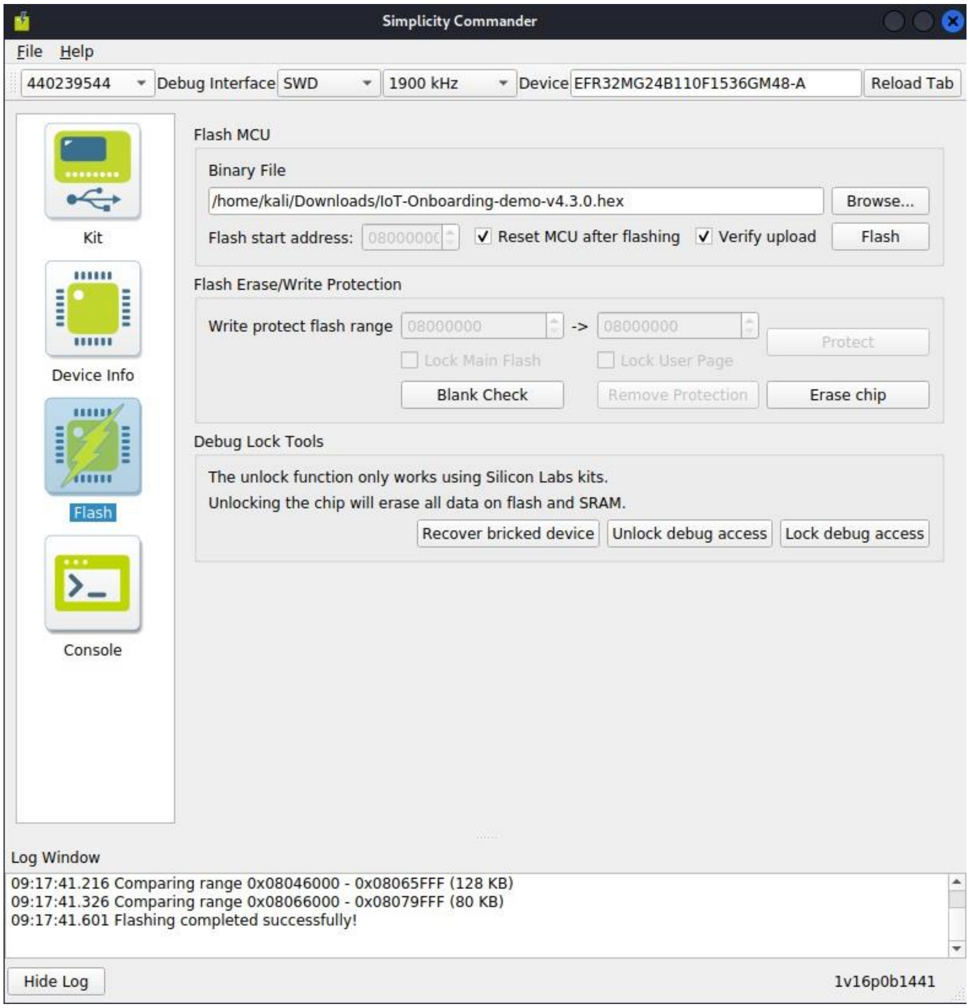
5.2 Silicon Labs Dev Kit (BRD2601A)

The Silicon Labs Dev Kit acts as the IoT device for this build. It is controlled using the Simplicity Studio v5 Software available at the official Simplicity Studio page and connected to a computer running Windows or Linux via USB. Our implementation leveraged a Linux machine running Simplicity Studio. Custom firmware for the Dev Kit leveraged in this use case was made by Silicon Labs.

5.2.1 Setup and Configuration

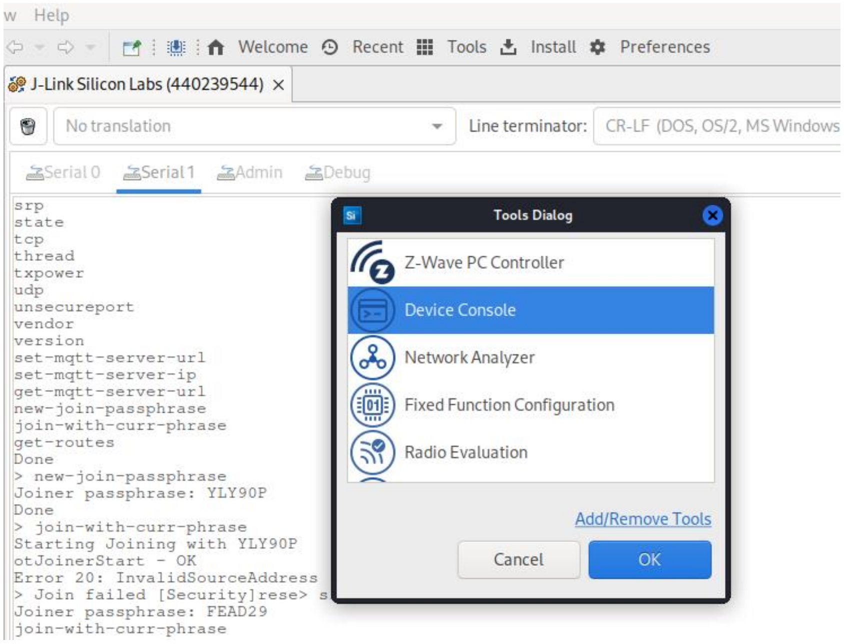
The Dev Kit custom firmware image works in conjunction with the Kudelski keySTREAM service. More information is available by contacting Silicon Labs through their contact form. Once the custom firmware has been acquired the Dev Kit can be configured using the following steps.

1. Connect the Dev Kit via USB to the machine running Simplicity Studio.
2. The firmware is installed onto the Dev Kit using the Simplicity Commander tool within Simplicity Studio.



After selecting the firmware file, click **Flash** to flash the firmware the Dev Kit.

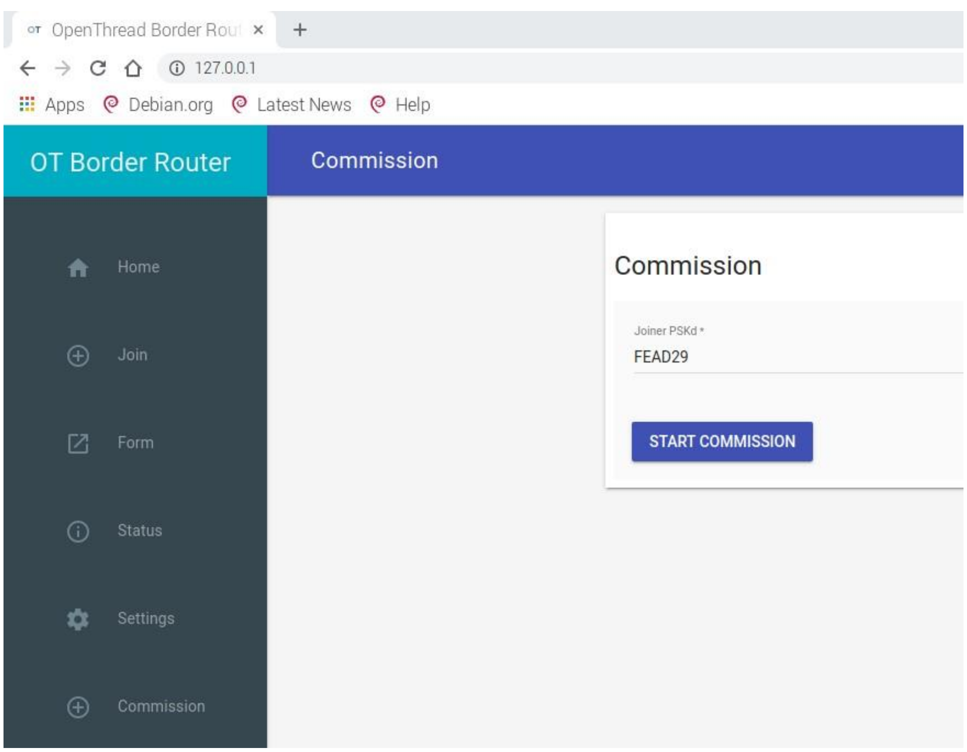
3. Open the device console in the **Tools** tab and then select the **Serial 1** tab.



4. Enter the following command to create a new join passphrase in the Serial 1 command line:

```
new-join-passphrase
```

5. Enter the output of the previous command in the **Commission** tab in the OpenThread Border Router GUI and click **Start Commission.**



6. In the Simplicity Commander Device Console, enter the following command to begin the joining process from the Thunderboard:

```
join-with-curr-phrase
```

7. Press the **Reset** button on the Dev Kit and the device will join the thread network and reach out to the Kudelski keySTREAM service. You should see the following output in the Simplicity Commander Device Console:

```
Joiner passphrase: FEAD29
join-with-curr-phrase
Starting Joining with FEAD29
otJoinerStart - OK
Error 20: InvalidSourceAddress
> Join successgot valid ext route
role changed to 2
coap start complete

kta_app start

Calling ktaInitialize
ktaInitialize Succeeded

Calling ktaStartup
ktaStartup Succeeded
KTA life cycle state --> INIT

Calling ktaSetDeviceInformation
ktaSetDeviceInformation Succeeded
KTA life cycle state --> SEALED

Calling ktaExchangeMessage
ktaExchangeMessage Succeeded
```

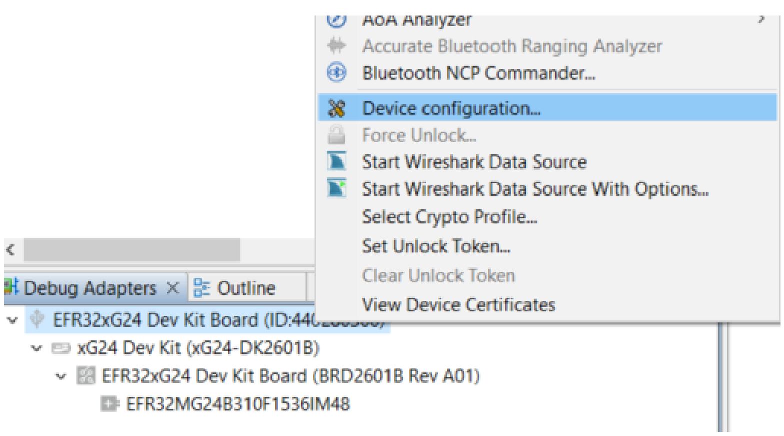
5.3 Kudelski keySTREAM Service

In this section we describe the Kudelski keySTREAM service which this build utilizes to provision certificates for connecting to the AWS IoT core. More information on keySTREAM is available at the keySTREAM page.

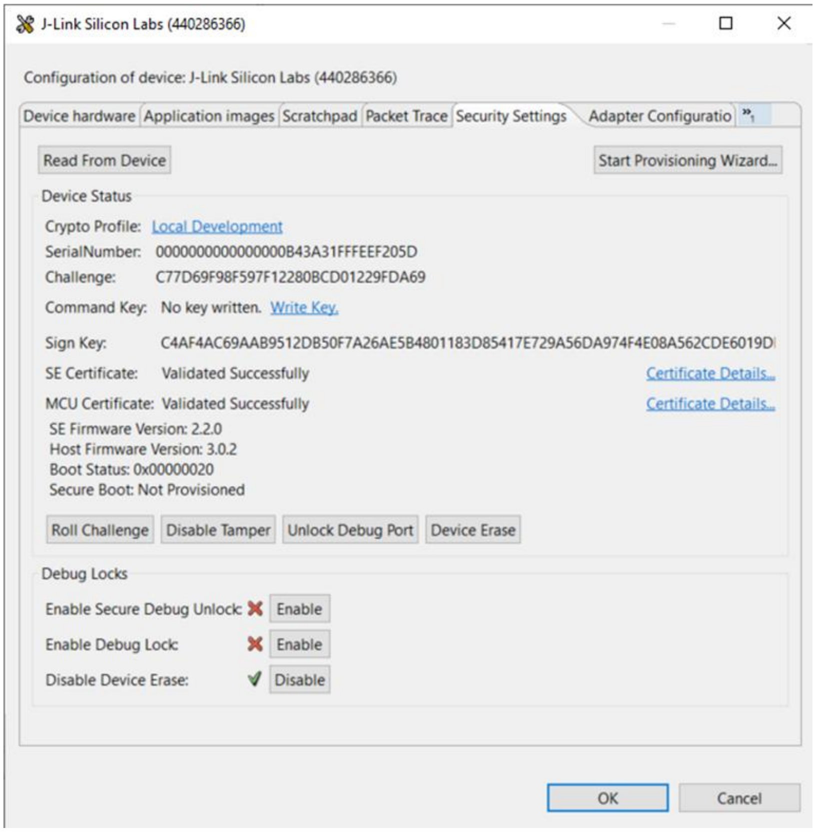
5.3.1 Setup and Configuration

The Kudelski keySTREAM service provides two certificates for the device: a CA certificate and a Proof of Possession (POP) certificate that is generated using a code from the AWS server. This section describes the steps to download these certificates.

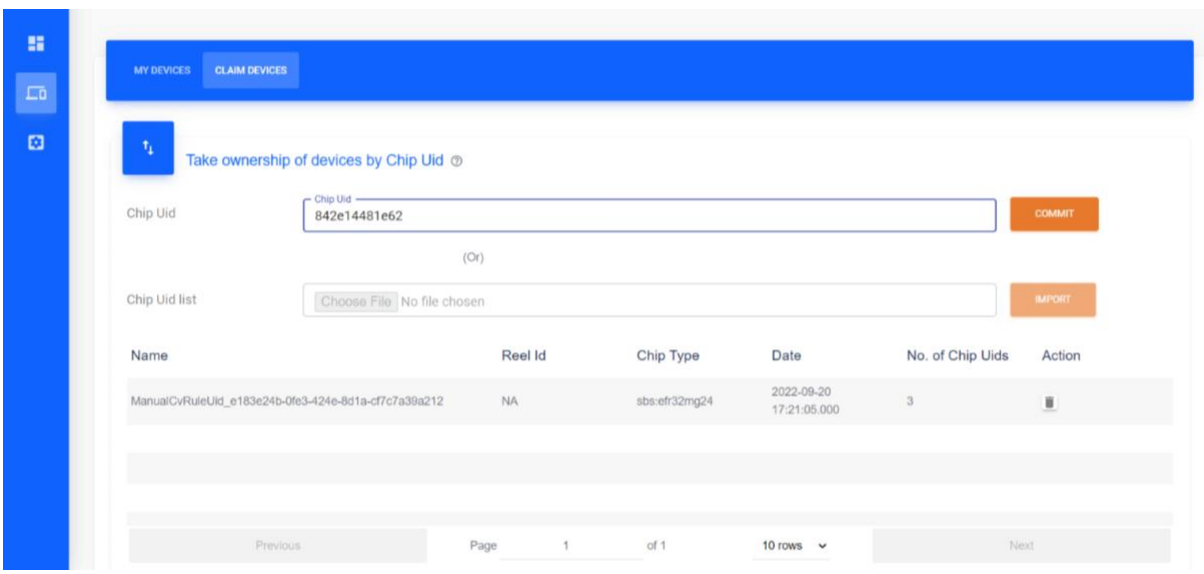
1. Locate the Chip UID for the Silicon Labs Dev Kit in Simplicity Studio by right clicking on the **Device Adapters** tab at the bottom and selecting **Device Configuration**.



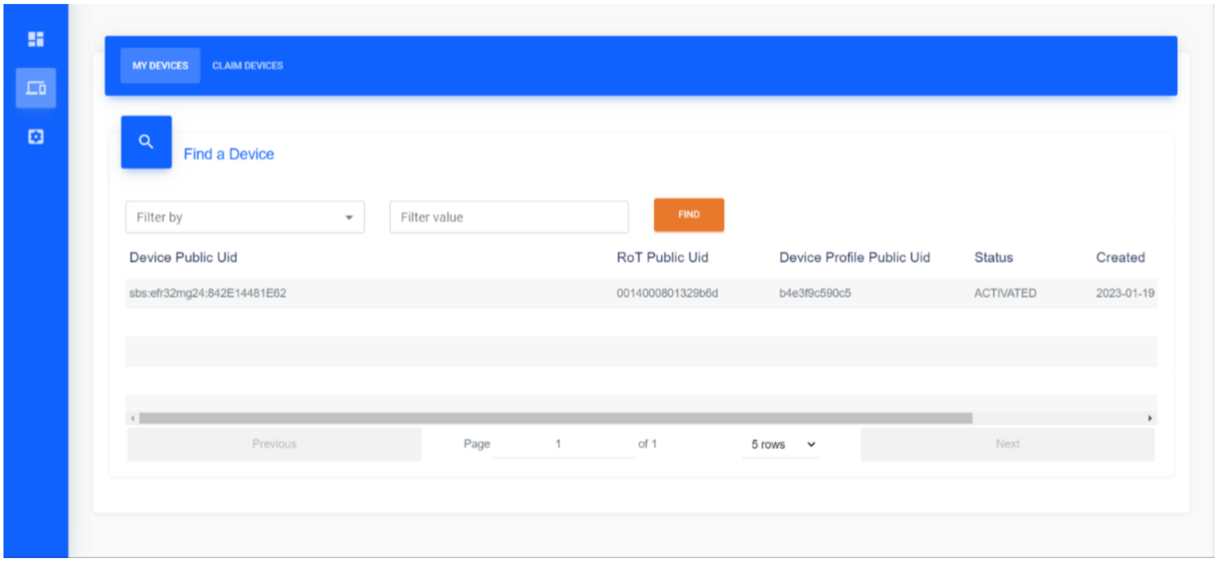
2. On the **Security Settings** tab, take the last 16 characters of the serial number and remove the 'FFFE' characters from the $7^{th} - 11^{th}$ positions.



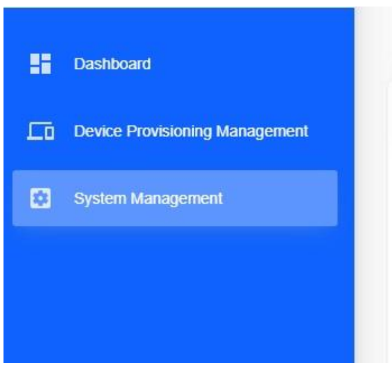
3. In the Kudelski keySTREAM service, claim your device by entering the chip UID from Simplicity Studio and clicking **Commit**.



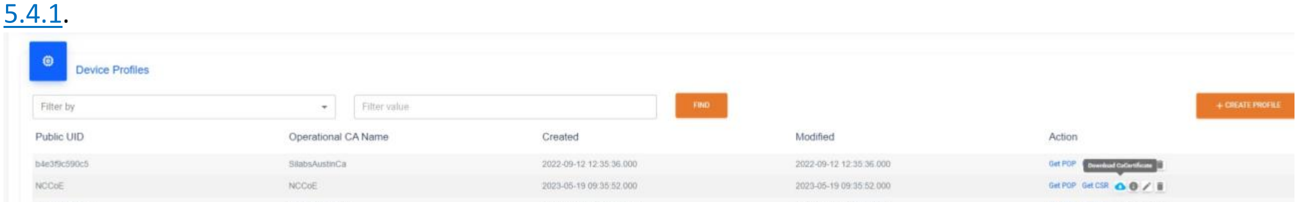
4. The device will now be visible in the **My Devices** tab. A device can be removed from the keySTREAM service by scrolling to the right and clicking the **Refurbish** button.



5. Open the **System Management** tab on the left side:



6. Click the cloud icon to download the CA Certificate and the POP certificate, the POP certificate will require a code that is obtained from the AWS IoT Core which will be generated in [Section 5.4.1](#).



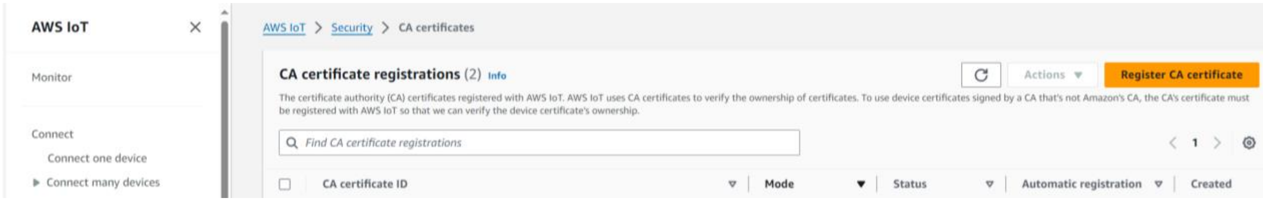
5.4 AWS IoT Core

The Silicon Labs Dev Kit will connect to the AWS MQTT test client using the certificates provisioned from the Kudelski keySTREAM service.

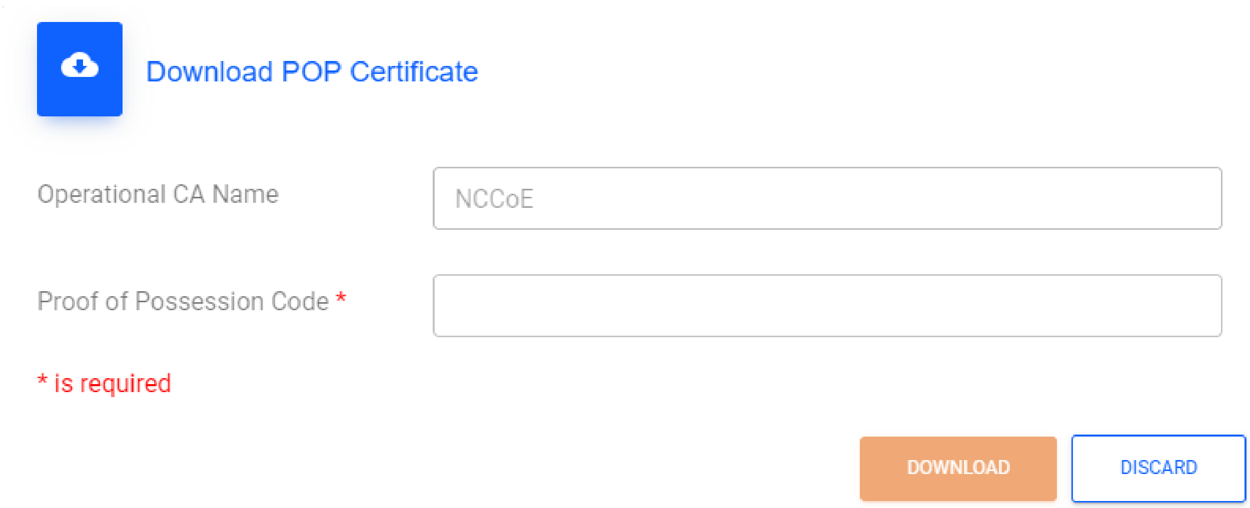
5.4.1 Setup and Configuration

Application-layer onboarding for this build is performed using the AWS MQTT test client. Certificates provisioned from the Kudelski keySTREAM service are uploaded to an AWS instance and the device will demonstrate its ability to successfully send a message to AWS.

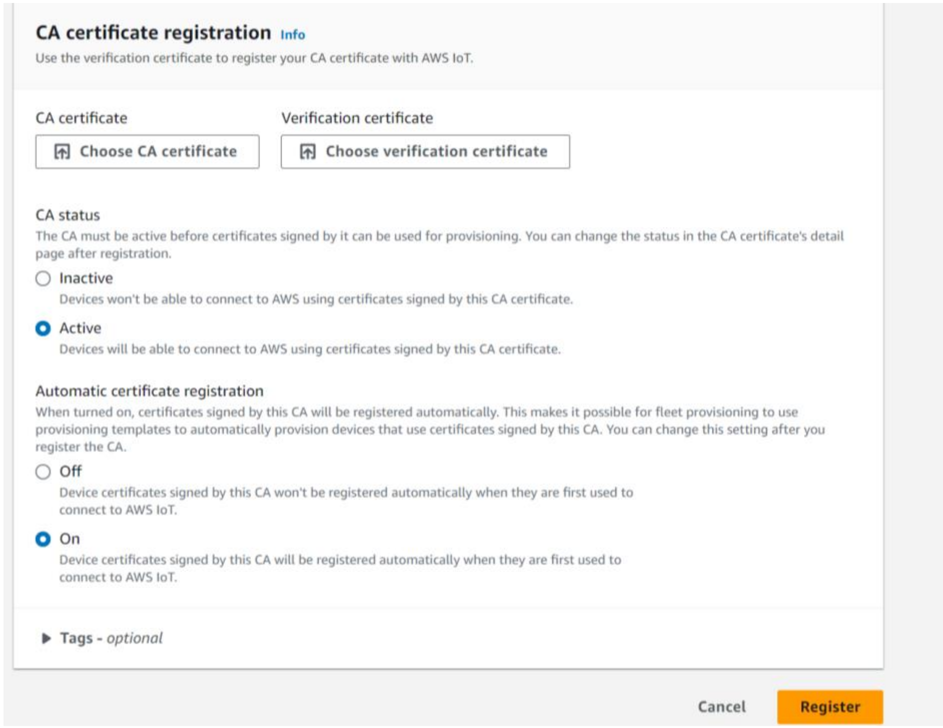
1. Within the AWS IoT Core, open the **Security** drop-down menu, click on **Certificate authorities**, and click the **Register CA certificate** button on the right.



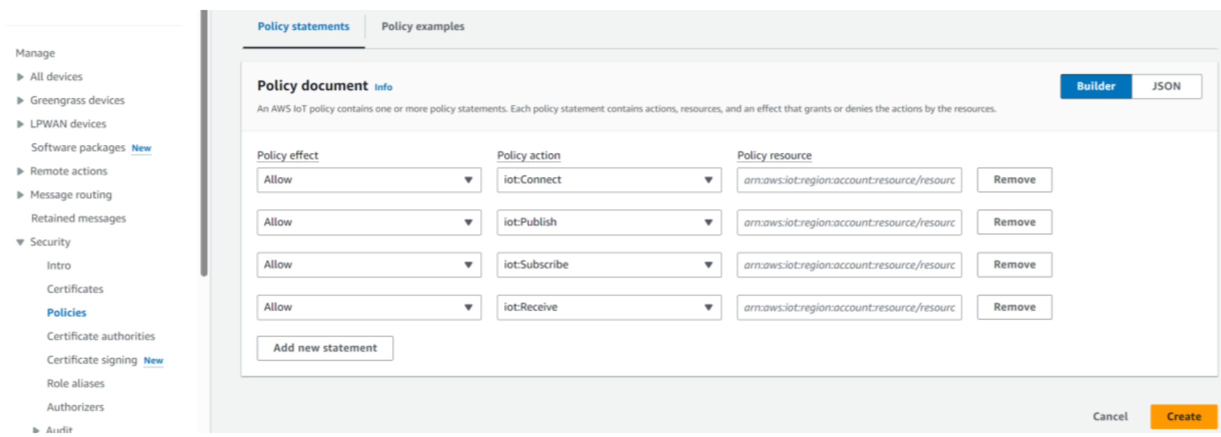
2. Select the radio button for **Register CA in Single-account mode** and copy the registration code to use as the **Proof of Possession Code** in the Kudelski keySTREAM service and download the POP certificate.



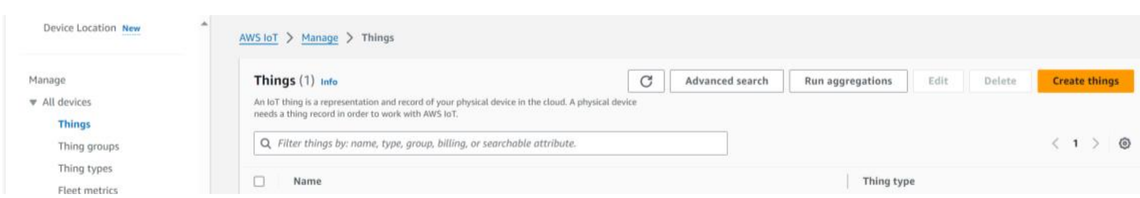
3. After downloading the POP certificate, upload the CA certificate and the POP (verification) certificate, and select the radio buttons for **Active** under **CA Status** and **On** under **Automatic Certificate Registration**. Then click **Register**.



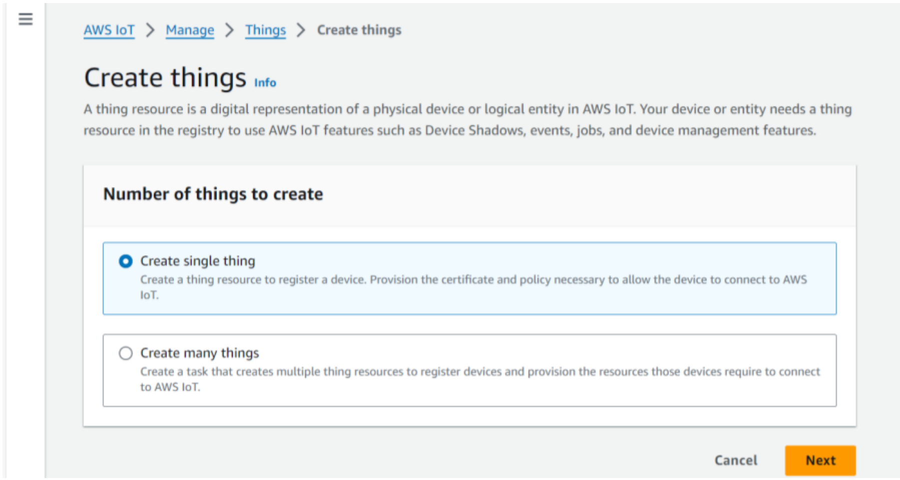
4. In the **Security** drop down menu, click on **Policies** and add the policies shown below. Then, click **Create**.



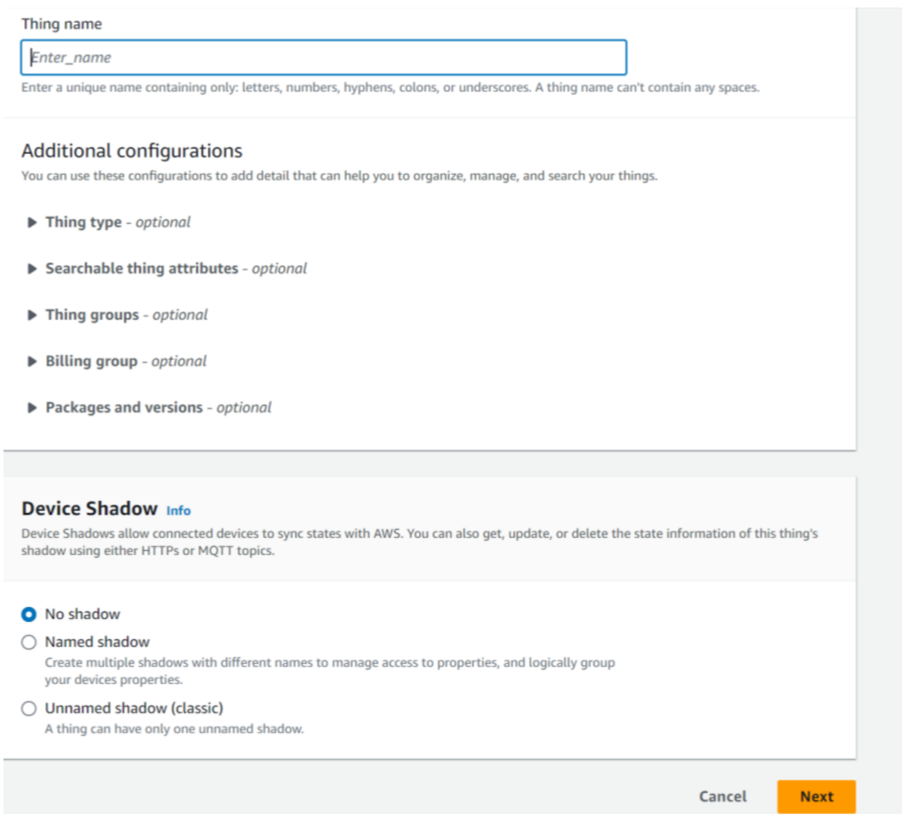
5. In the **All devices** drop-down menu, click on **Things** and click **Create things**.



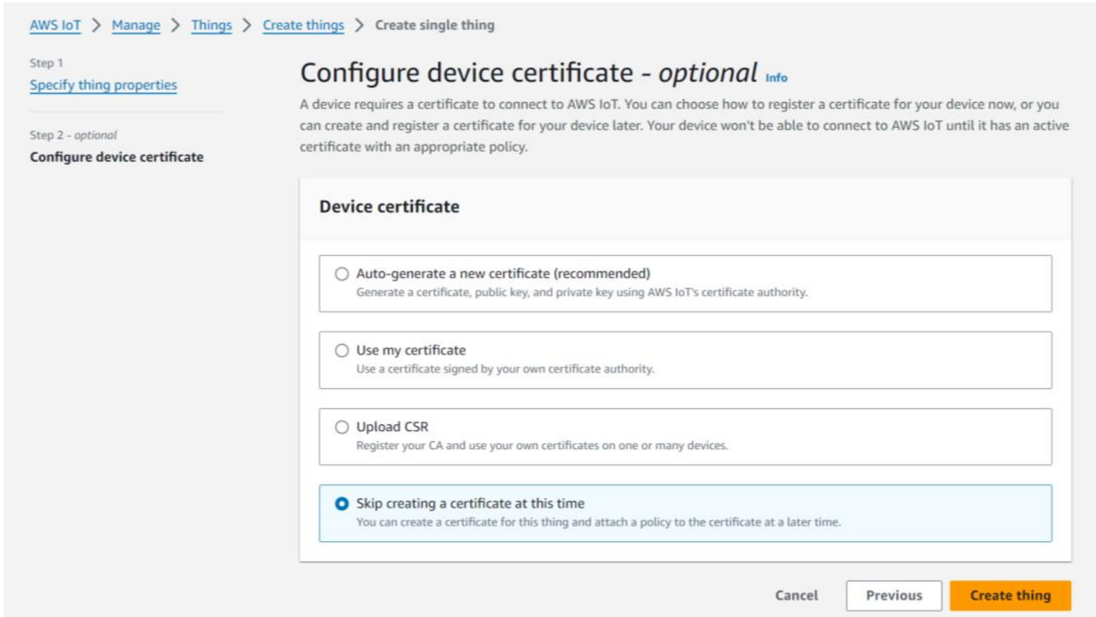
6. Click the **Create single thing** radio button and click **Next**.



7. Enter a **Thing name** and click **Next.**

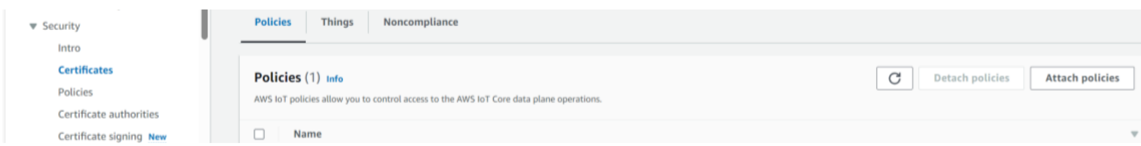


8. Select the **Skip creating a certificate at this time** radio button and click **Create thing**.

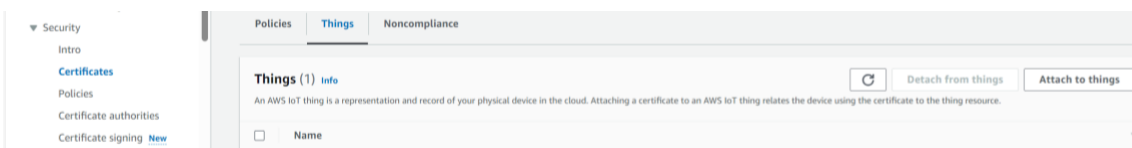


9. In the **Security** drop-down menu, click on **Certificates** and click the Certificate ID of the certificate that you created.

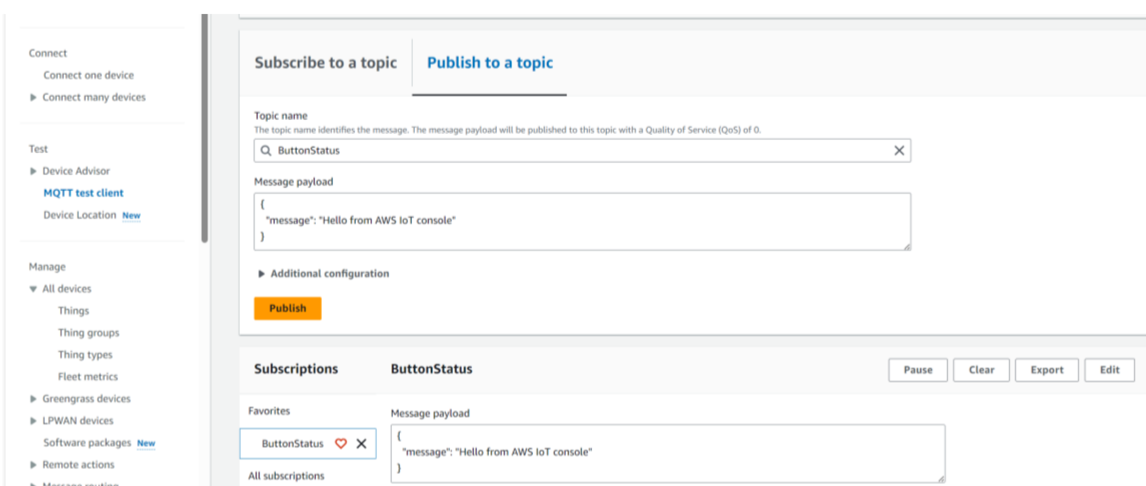
10. In the **Policies** tab at the bottom, click **Attach policies** and add the policy that you created.



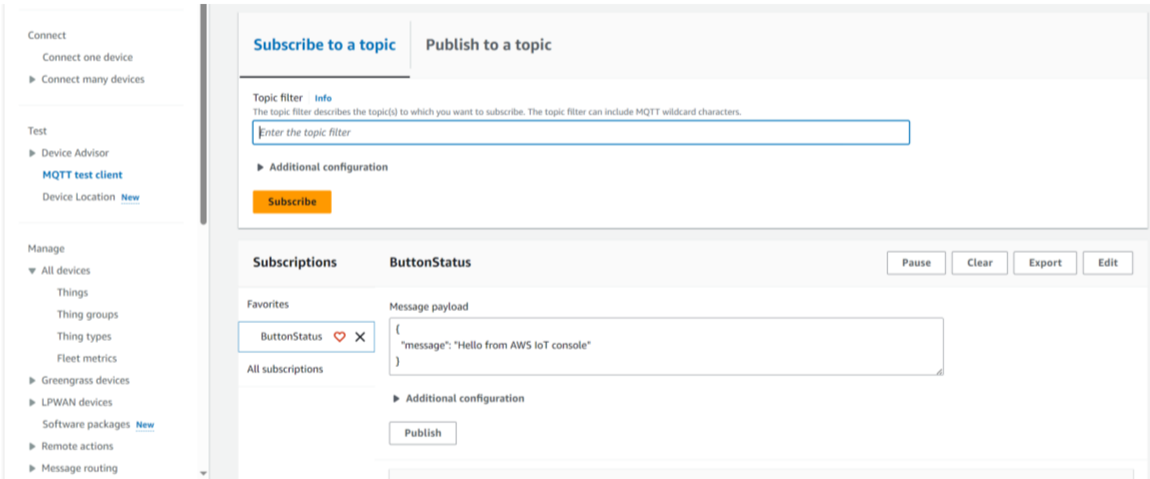
11. In the **Things** tab, click **Attach to things** and add the thing that you created.



12. Click the **MQTT test client** on the left side of the page and click the **Publish to a topic** tab.



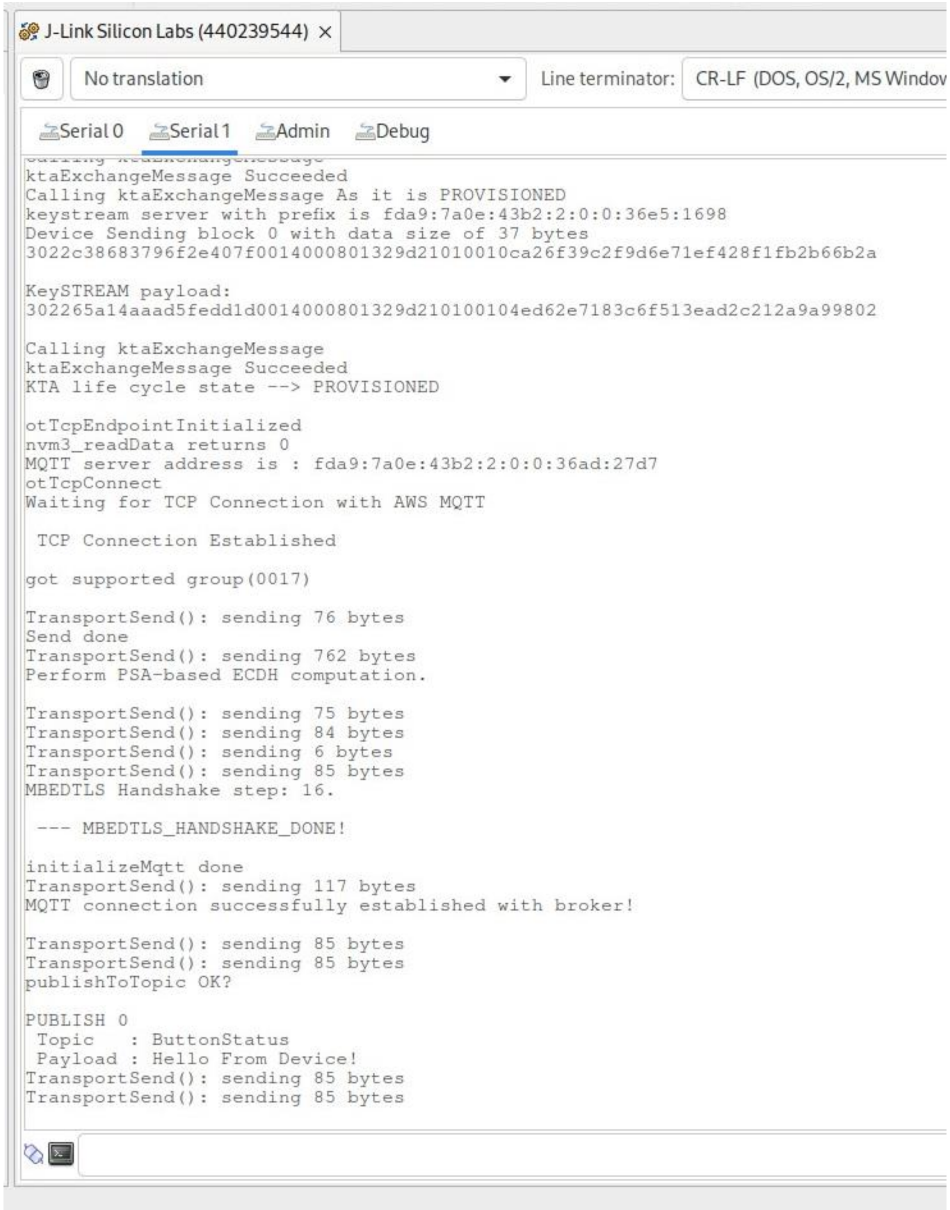
13. Create a message of your choosing and click **Publish**. On the **Subscribe to a topic** tab, make sure that you are subscribed to the topic that you just created.



5.4.2 Testing

Information sent and received by the Silicon Labs Dev Kit to the MQTT test client will be displayed in the device console in Simplicity Commander. This section describes testing the communication between the MQTT test client and the device.

1. On the Thunderboard, press **Button 0**. This will begin the connection to the MQTT test client.



6 Build 5 (BRSKI over Wi-Fi, NquiringMinds)

This section of the practice guide contains detailed instructions for installing and configuring all of the products used to build an instance of the example solution. For additional details on Build 5's logical and physical architectures, see NIST SP 1800-36B: *Approach, Architecture, and Security Characteristics*.

The network-layer onboarding component of Build 5 utilizes the BRSKI protocol.

6.1 Pledge

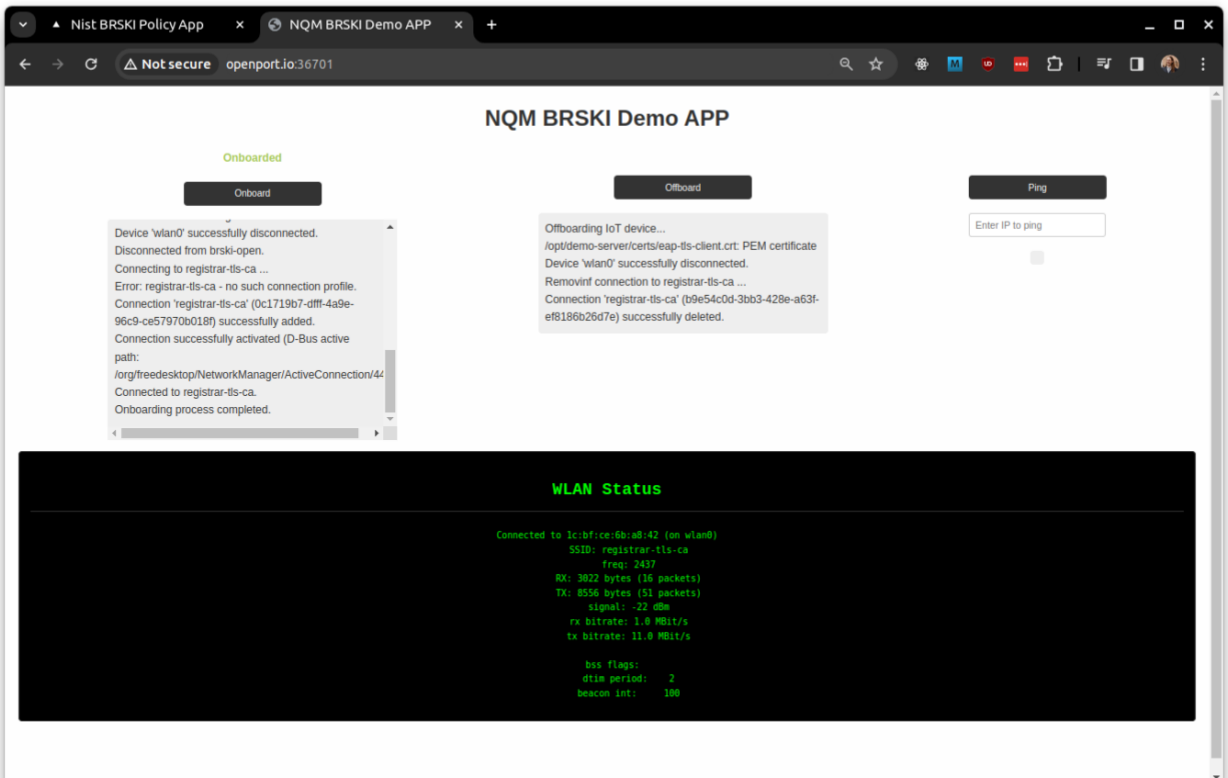
The Pledge acts as the IoT device which is attempted to onboard onto the secure network. It implements the pledge functionality as per the IETF BRSKI specification. It consists of software provided by NquiringMinds running on a Raspberry Pi Model 4B.

6.1.1 Installation and Configuration

Hardware requirements, pre-installation steps, installation steps, and configuration [instructions for the pledge device] can be found at the official NquiringMinds repository.

6.1.2 Operation and Demonstration

To demonstrate the onboarding and offboarding functionality, NquiringMinds has provided a web application which runs on the pledge device. It features a button one can use to manually run the onboarding script and display the output of the onboarding process, as well as a button for offboarding. It also features a button to ping an IP address, which is configured to ping the designated address via the wireless network interface.



6.2 Router and Logical Services

The router and logical services were hosted on a Raspberry Pi Model 4B equipped with 2 external Wi-Fi adapters. These additional Wi-Fi adapters are needed to support VLAN tagging which is a hardware dependent feature. The [details of the physical setup and all connections] are provided in the official NquiringMinds documentation.

6.2.1 Installation and Configuration

All of the services described in the next section can be installed on a Raspberry Pi using the [installer provided by NquiringMinds].

The demonstration services can also be built from source code, if needed. The following links provide the instructions for building each of those services:

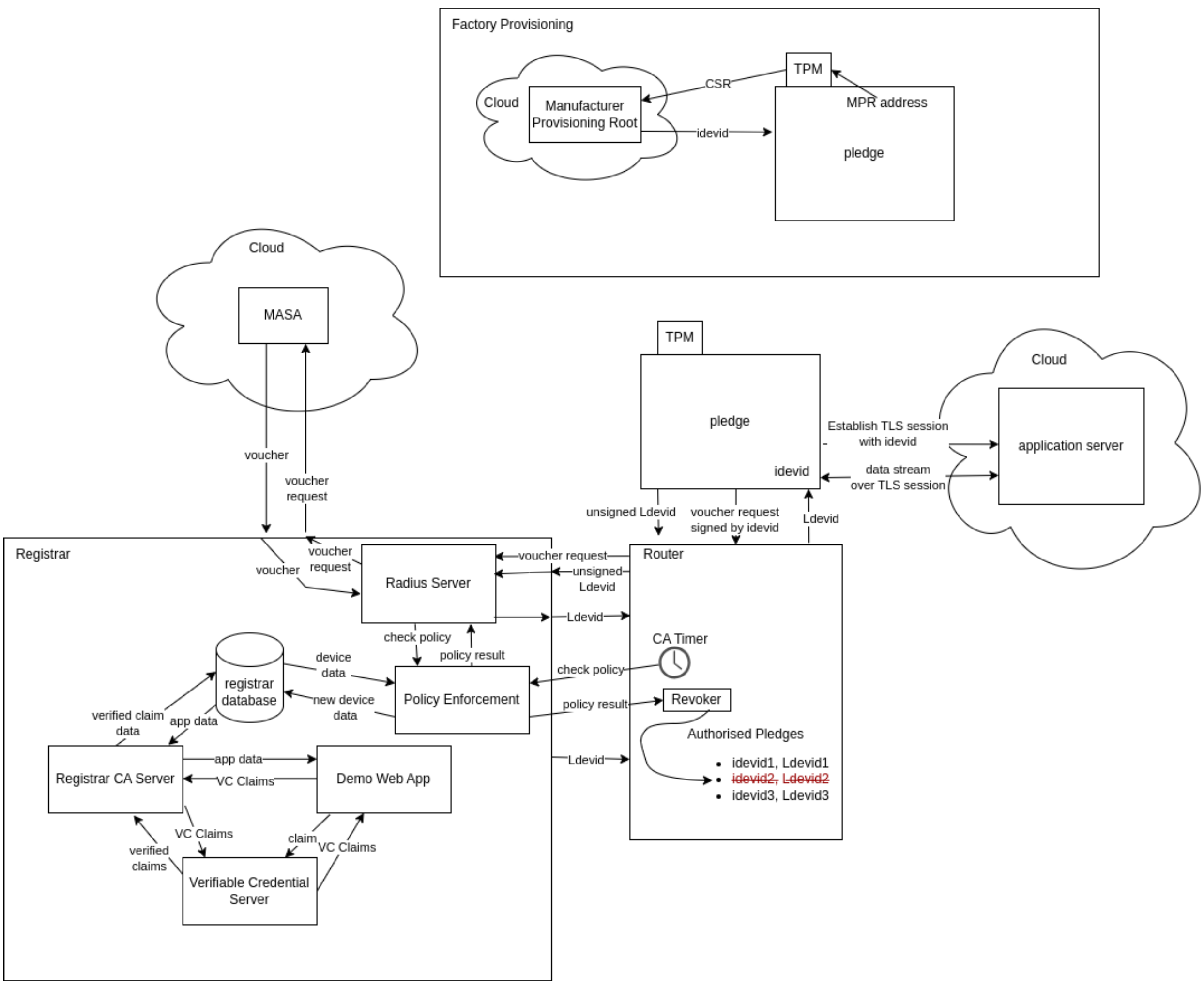
- BRSKI Demo Setup
- EAP Config
- MDNS publishing services

6.2.2 Logical services

The following logical services are installed on the Registrar and services device. The implementation of these services are to be found at the following repository links: NIST BRSKI implementation and BRSKI.

Figure 6-1 below describes how these entities and logical services fit together to perform the BRSKI flow, and a top-level view of how information is transmitted throughout the services to onboard the pledge.

Figure 6-1 Logical Services for Build 5



6.2.2.1 MASA

The MASA currently resides as a local service on the registrar. In practice, this service would be located on an external server managed by the manufacturer. The MASA verifies that the IDevID is authentic, and that the IDevID was produced by the manufacturer's MPR.

6.2.2.2 Manufacturer Provisioning Root (MPR)

The MPR sits on an external server and provides the IDevID (X.509 Certificate) for the device to initialize it after production and notarize it with a unique identity. The address of the MPR is built into the firmware of the device at build time.

6.2.2.3 Registrar

Build 5's BRSKI Domain Registrar runs the BRSKI protocol modified to work over Wi-Fi and functions as the Domain Registrar to authenticate the IoT devices, receive and transfer voucher requests and responses to and from the MASA and ultimately determines whether network-layer onboarding of the device is authorized to take place on the respective network. NquiringMinds has developed a stateful non-persistent Linux app for android that serves this purpose.

The registrar is responsible for verifying if the IDevID certificate provided by the pledge is authentic, by verifying it with the MASA and verifying that the policy for a pledge to be allowed onto the closed secure network has been met. It also runs continuous assurance periodically to ensure that the device still meets the policy requirements, revoking the pledge's access if at a later time it doesn't meet the policy requirements. Signed verifiable credential claims may be submitted to the registrar to communicate information about entities, which it uses to update its database used to determine if the policy is met, the tdx Volt is used to facilitate signing and verification of verifiable credentials. In the demonstrator system the MASA and router are integrated into the same physical device.

6.2.2.3.1 Radius server (Continuous Assurance Client)

To provide continuous assurance capabilities for connected IoT devices, the registrar includes a Radius server that integrates with the Continuous Assurance Server.

The continuous assurance policy is enforced by a script which periodically runs to check that the policy conditions are met. It accomplishes this by querying the Registrar's SQLite database. For the demonstration, the defined policy is:

- The manufacturer and device must be trusted by a user with appropriate privileges
- The device must have a device type associated
- The vulnerability score of the SBOM for the device type must be lower than 6
- The device must not have contacted a denylisted IP address within the last 2 minutes

If the device fails any of these checks, the device will be offboarded.

6.2.2.4 Continuous Assurance Server

The registrar runs several services used to power the continuous assurance flow.

6.2.2.4.1 Verifiable Credential Server

The verifiable credential server is used to sign verifiable credentials submitted through the Demo web app and verify verifiable credentials submitted to the registrar, it is powered by the functionality of the tdx Volt, a local instance of which is run on the registrar.

The code for the Verifiable Credential Server is hosted at the GitHub repository.

6.2.2.4.2 Registrar Continuous Assurance Server

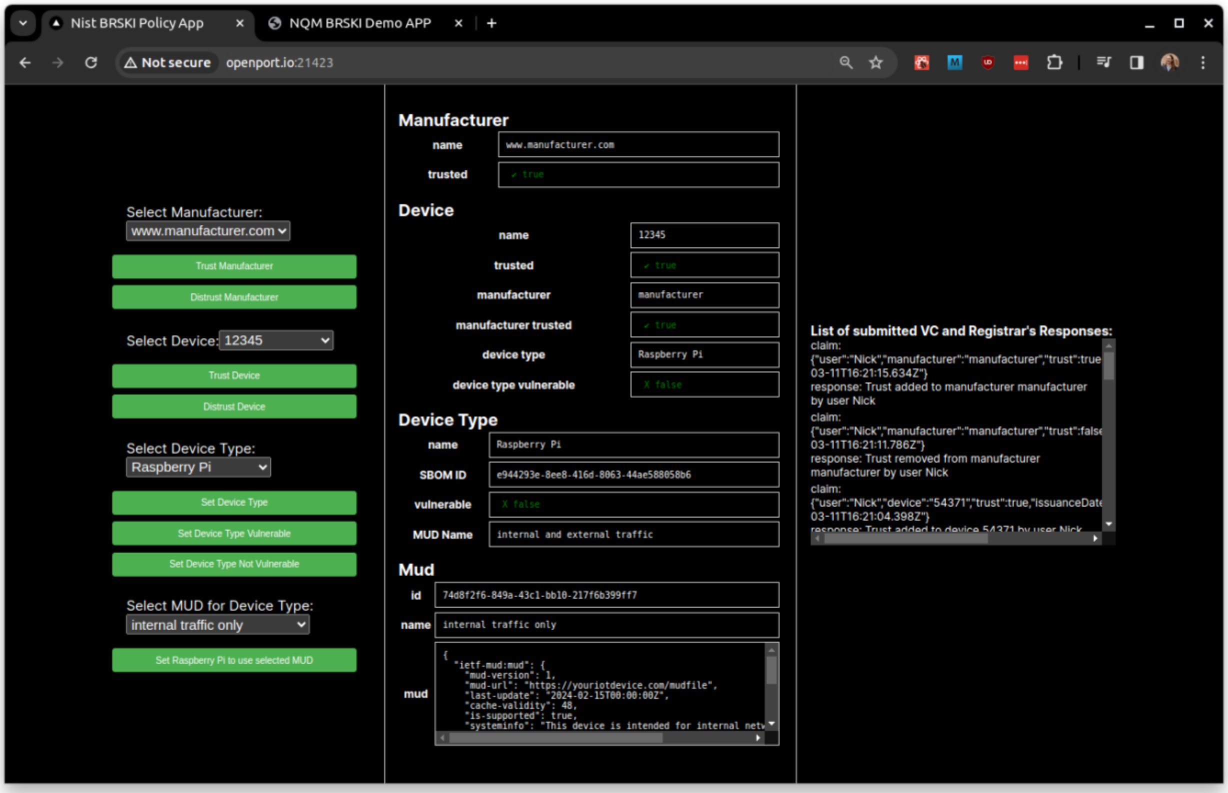
The registrar hosts a REST API which is used to interface with the registrar's SQLite database which stores information about the entities the registrar knows of. This server utilizes the verifiable credential server to verify submitted verifiable credential claims submitted to it.

The code for the Registrar Continuous Assurance Server is hosted at the GitHub repository.

6.2.2.4.3 Demo Web Application

The demo web application is used as an interactive user-friendly way to administer the registrar. Users can view the list of verifiable credentials submitted to the registrar. The application also displays the state of the manufacturers, devices, device types and Manufacturer Usage Description (MUD). There are buttons provided which allow you to trust or distrust a manufacturer, trust or distrust a device, set the device type for a device, set if a device type is vulnerable or not and set the MUD file associated with the device type. All of these operations are performed by generating a verifiable credential containing the claim being made, which is then submitted to the verifiable credential server to sign the credential. The signed verifiable credential is then sent to the registrar continuous assurance server to be verified and used to update the SQLite database on the registrar.

The code for the Demo Web Application is hosted at the GitHub repository.



6.2.2.5 Application server

The application server sits on a remote server and represents the server for an application which should consume data from the pledge device. The pledge device uses the IDevID certificate to establish a secure TLS connection to onboard onto the application server and begin sending data autonomously, currently OpenSSL s_client is used from the pledge to establish a TLS session with the application server, running on a server off-site, and the date and CPU temperature are sent to be logged on the application server, as a proof of principle.

6.2.2.5.1 Installation/Configuration

Hardware requirements, pre-installation steps, installation steps, and configuration [instructions for the router]() can be found at the official NquiringMinds repository.

6.2.2.5.2 Operation/Demonstration

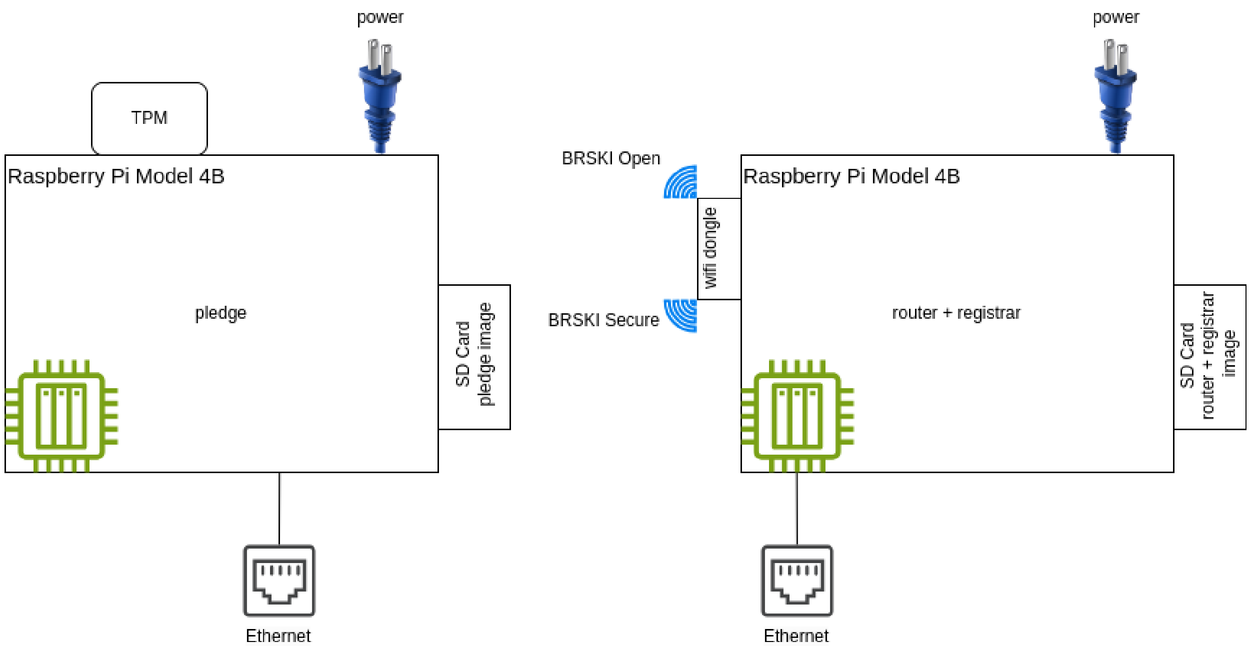
The instructions to use this factory use case code to provision an IDevID onto your pledge are also located at the official NquiringMinds repository in the above section.

6.3 Onboarding Demonstration

6.3.1 Prerequisites

Prior to beginning the demonstration, the router and pledge devices must be connected to power, and to the network via their ethernet port. On boot, both devices should start the services required to demonstrate the BRSKI flow.

Figure 6-2 Diagram of Physical/Logical Components Used to Demonstrate BRSKI Flow



To support the demo and debug features the pledge and the registrar need to be connected to physical ethernet, ideally with internet access. They should still function without an internet connection, but the vulnerability scores of the SBOMs will not be updated and the demo web apps will only be accessible on the local network.

The detailed networking setup details are available in the NquiringMinds NIST Trusted Onboarding Build-5.

6.3.2 Onboarding Demonstration

Once configuration of the devices and the prerequisite conditions have been achieved, the onboarding demonstration can be executed following NquiringMinds Demo Continuous Assurance Workflow.

6.3.3 Continuous Assurance Demonstration

The instructions to demonstrate the continuous assurance workflow are contained in the official NquiringMinds documentation.

6.4 BRSKI Factory Provisioning Build

This Factory Provisioning Build includes many of the components listed in Section 6.2, including the Pledge, Registrar, and other services. An Infineon Secure Element was also included in the build and provides secure generation and storage of the key material and certificates provisioned to the device.

6.4.1 Pledge

The Pledge acts as the IoT device which is attempting to onboard onto the secure network. It implements the pledge functionality as per the IETF BRSKI specification. It consists of a Raspberry Pi Model 4B equipped with an Infineon Optiga SLB 9670 TPM 2.0 Secure Element. The Infineon Secure Element was connected to a Raspberry Pi via the built-in GPIO pins present on the Pi.

6.4.1.1 Factory Use Case - IDevID provisioning

NquiringMinds provided demonstration code that generates a public/private keypair within the secure element, creates a CSR, and uses that CSR to obtain an IDevID certificate from tdx Volt. The demonstration process can be found at the official NquiringMinds documentation.

Initially, it generates a CSR using the TPM secure element to sign it, it then sends the CSR to the MPR server which is the manufacturer's IDevID Certificate Authority and is bootstrapped in the vanilla firmware on the pledge's creation in the factory. The MPR sends back a unique IDevID for the pledge which it stores in its secure element.

The code for this is hosted at the official NquiringMinds repository.

6.4.2 Installation and Configuration

Hardware requirements, pre-installation steps, installation steps, and configuration instructions for the pledge can be found at the official NquiringMinds repository referenced above.

6.4.3 Operation and Demonstration

The instructions to use this factory provisioning use case code to provision an IDevID onto the pledge is also located in the official NquiringMinds repository referenced above.

Appendix A List of Acronyms

AKM	Authentication and Key Management
AOS	ArubaOS
AP	Access Point
API	Application Programming Interface
ASN.1	Abstract Syntax Notation One
AWS	Amazon Web Services
BRSKI	Bootstrapping Remote Secure Key Infrastructure
BSS	Basic Service Set
CA	Certificate Authority
CRADA	Cooperative Research and Development Agreement
CSR	Certificate Signing Request
DMZ	Demilitarized Zone
DPP	Device Provisioning Protocol (Wi-Fi Easy Connect)
EAP	Extensible Authentication Protocol
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
HPE	Hewlett Packard Enterprise
IaaS	Infrastructure as a Service
IDevID	Initial Device Identifier
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
LDevID	Locally Significant Device Identifier
MASA	Manufacturer Authorized Signing Authority
MPR	Manufacturer Provisioning Root
MUD	Manufacturer Usage Description
MQTT	MQ Telemetry Transport

NCCoE	National Cybersecurity Center of Excellence
NIST	National Institute of Standards and Technology
OCF	Open Connectivity Foundation
OS	Operating System
OTBR	Open Thread Border Router
PNG	Portable Network Graphics
POP	Proof of Possession
QR	Quick-Response
RF	Radio Frequency
SBOM	Software Bill of Materials
SP	Special Publication
SoC	System-on-Chip
SSID	Service Set Identifier
TPM	Trusted Platform Module
UID	Unique Identifier
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UXI	User Experience Insight
VLAN	Virtual Local Area Network
VM	Virtual Machine
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3

Appendix B References

[1] Wi-Fi Alliance. *Wi-Fi Easy Connect*. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-easy-connect.