AN3170

Getting Started with AWS IoT Greengrass® on SAMA5D2

Introduction

This application note shows how to implement a Cloud Edge Node utilizing a Microchip SAMA5D2 MPU, a Microchip Secure Hardware Element, and Amazon Web Services (AWS®) IoT Greengrass.

What are Cloud Services?

Cloud services provide resources that are accessible over the Internet. These available resources are not strictly limited, and are increasing in number and functionality almost daily. One of the main benefits of Cloud services is removing the burden of maintaining compute and storage servers by individuals and organizations. The servers that implement Cloud services are maintained by service providers such as Amazon Web Services (AWS). Internet of Things (IoT) is a subset of Cloud services that are tailored to devices such as actuators and sensors. An optimization of IoT services is a concept called Cloud Edge Computing. Edge computing brings web services to servers and devices located near the clients that are using those services. Although not owned by the Cloud service providers, the Edge Devices are under the control of the Cloud service providers.

Amazon Cloud Services

AWS IoT Greengrass is a combination of software and Cloud services that allow special AWS IoT devices to provide IoT services to other devices on the same local network. Not all AWS IoT devices can work with AWS IoT Greengrass. The SAMA5D2, however, can run AWS IoT Greengrass software and does provide the necessary compute resources listed in the AWS IoT system requirements.

Microchip and Edge Computing

In addition to the SAMA5D2, the system described in this app note includes an ATECC608A secure element. This secure element is utilized by Greengrass to implement Hardware Security Integration (HSI).

Table of Contents

1. Prerequisites

- SAMA5D27-WLSOM1-EK development board – Part number DM320117
- USB to UART cable – Part number FTDITTL-232R-3V3
- Linux® hostPC
- Micro USB cable
- Router connected to the Internet

2. Procedure Overview

The process of building an AWS IoT Greengrass system is straightforward, but requires several different activities. This application note divides the activities into easy-to-follow processes.

1. Set up the hardware. Refer to section 2.1 Setting Up the Hardware.
2. Set up AWS IoT services. Refer to section 2.2 Setting Up AWS Services.
3. Build the target image. Refer to section 4.1 Building the Target Image.
4. Configure the target. Refer to section 4.3 Copy Certificates and Root CA.
5. Run Greengrass on the target. Refer to section 4.7 Start the Greengrass Core.
6. Deploy a Greengrass group from the Cloud. Refer to section 4.8 Deploy the Group from AWS Console.

2.1 Setting Up the Hardware

Follow the steps below to set up the hardware:

1. Connect the Ethernet (or Wi-Fi®) cable from the board to a router with a live Internet connection. Refer to section 4.5 SAM5D27-WLSOM1-EK Setup for the Buildroot Image.
2. Connect the USB to UART cable (J26) to the host PC and to the debug connector on the SAMA5D27-WLSOM1-EK board.
3. Connect the micro USB cable from the host PC to the SAMA5D27-WLSOM1-EK (J10) to supply the board.
4. Plug the SD card image with Microchip Linux distribution.
5. Press the “nStrat_SOM” button on the board.

2.2 Setting Up AWS Services

Amazon Web Services (AWS) are a collection of Cloud services available for many different purposes. This application note uses a small subset of these services to get started using AWS IoT Greengrass. To get a Greengrass system working properly, several different AWS Services must be configured correctly to work together. These services include AWS IAM, AWS IoT Core, and AWS IoT Greengrass. One of the more tricky pieces of the system is getting the correct roles and permissions.

2.2.1 Overview

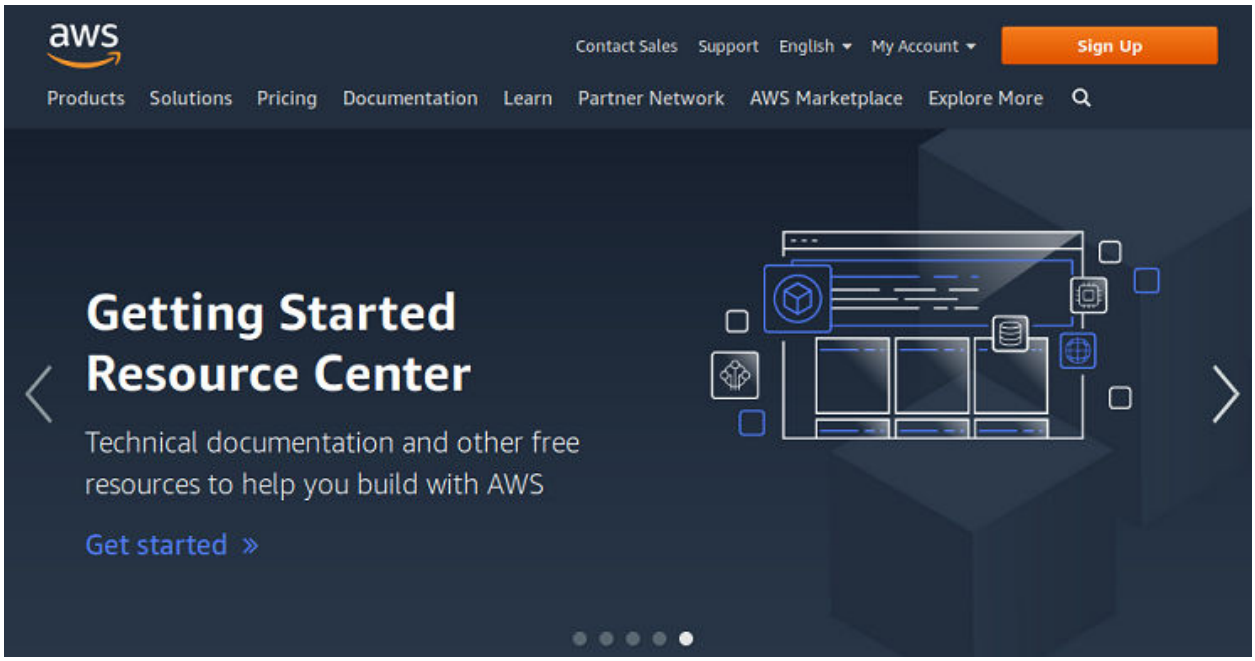
The process for setting up AWS services consists of the steps listed below. Details of each step are given in the sections that follow.

1. Create an AWS account if you do not already have one.
2. Open the AWS console.
3. Go to the correct region.
4. Create Greengrass objects – Group, Core, Certificates, Roles, and Permissions.
5. Download all certificates and keys.

2.2.2 Create an AWS Account

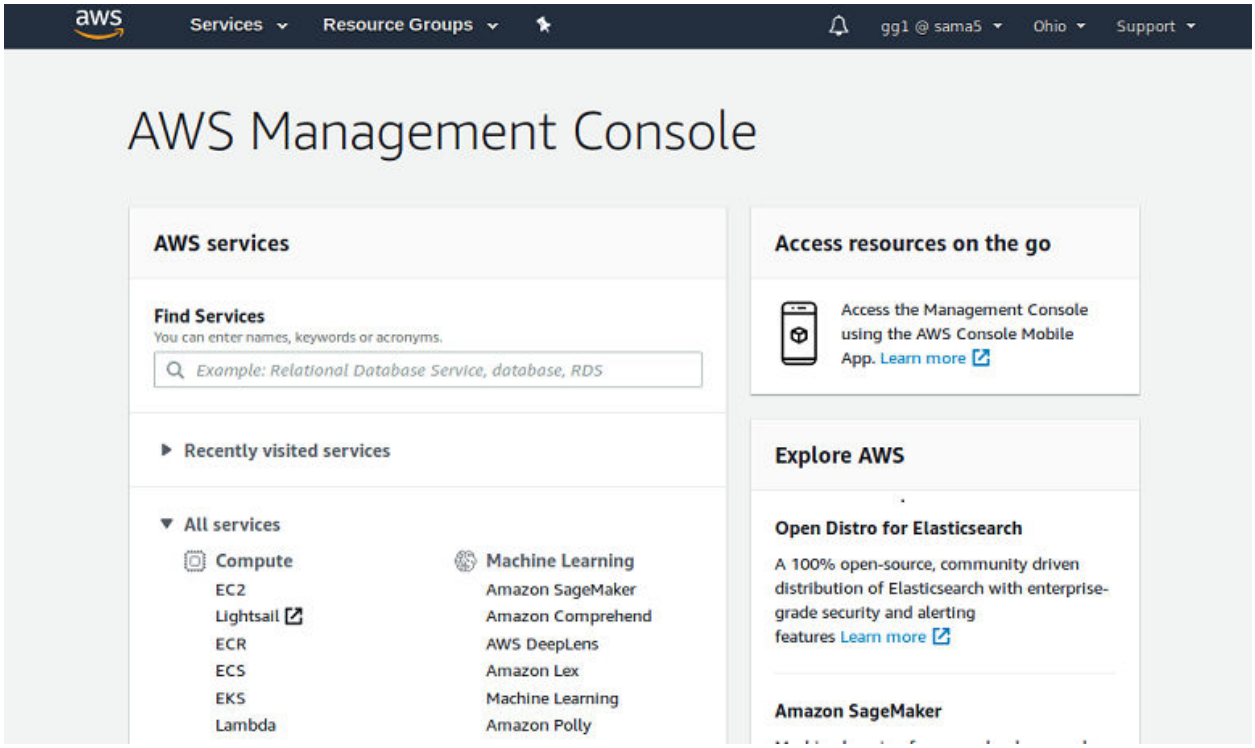
Open the AWS home page at http://aws.amazon.com/, and choose **Create an AWS Account**.

Follow the instructions after pressing the “Sign Up” button in the upper right.



2.2.3 Open the AWS Console

Go to the AWS console by going to https://console.aws.amazon.com/.



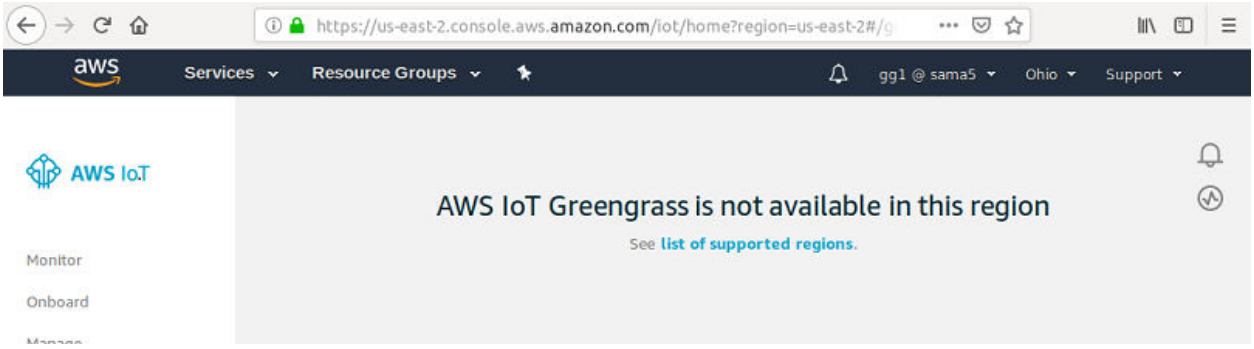
Search for Greengrass in the “Find Services” area. Then click on “IoT Greengrass”. Amazon Web Services are divided into regions. Not all regions support all AWS services. See the table below for the regions that support AWS IoT Greengrass.

When implementing a Greengrass system, be sure to use one of the regions listed below.

Table 2-1. Greengrass Regions

Region Name	Region
US East (N. Virginia)	us-east-1
US West (Oregon)	us-west-2
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
EU (Frankfurt)	eu-central-1
EU (Ireland)	eu-west-1

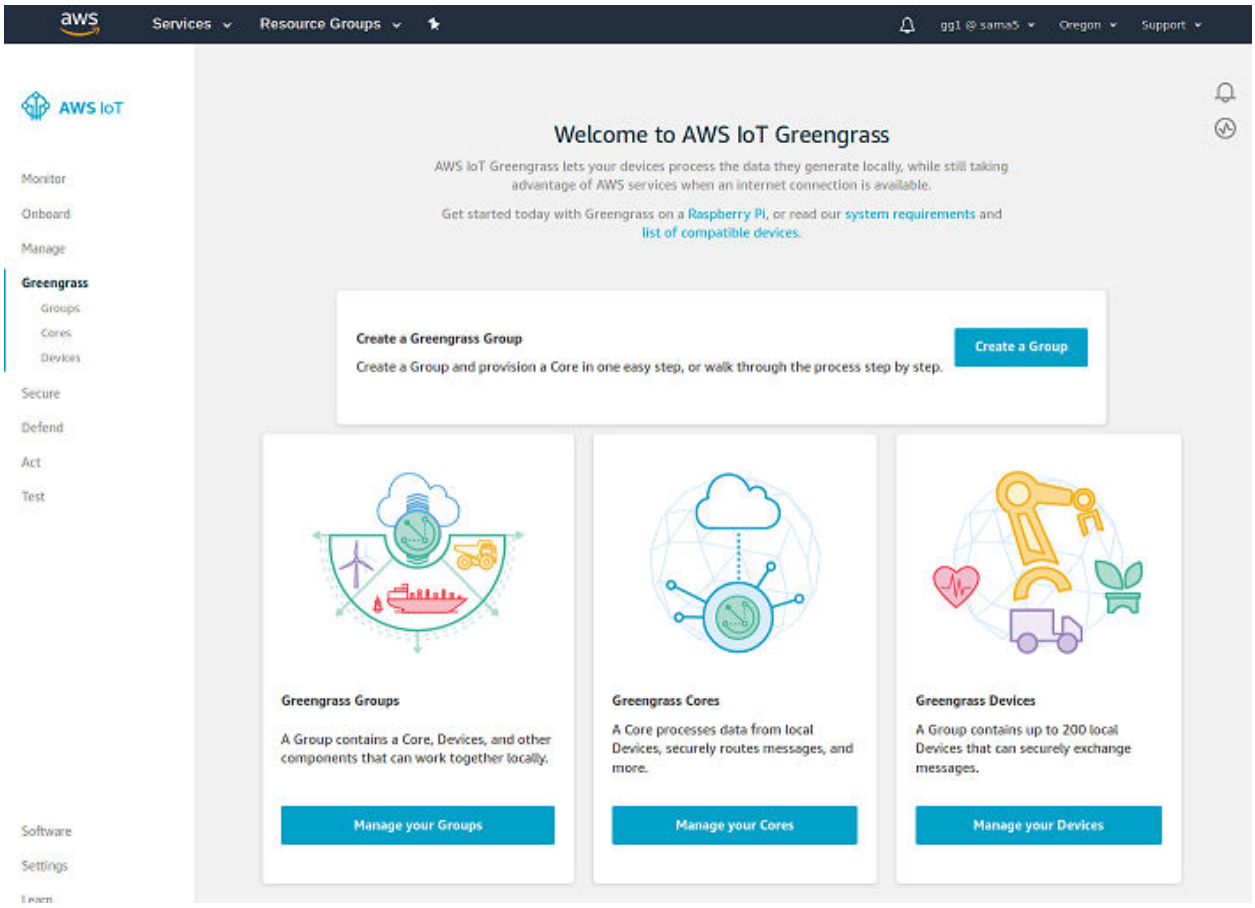
If the currently selected region does not support Greengrass, the following message will appear. Select a proper region by using the drop-down menu near the upper right.



2.2.4 Create a Greengrass Group

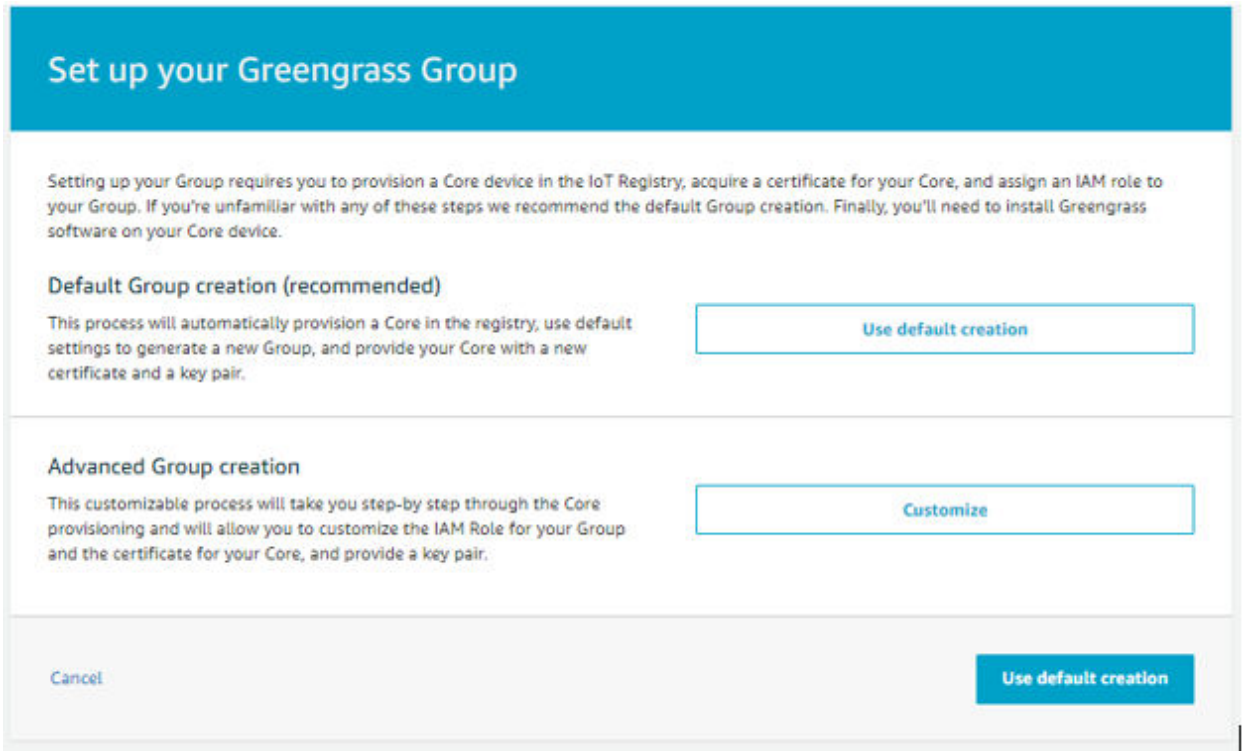
The main AWS console window for AWS IoT is shown below.

From the left navigation bar, select the “Greengrass” option if it is not already selected. Press the “Create a Group” button. A Greengrass group is a collection of items that are required to deploy a running Greengrass system.



2.2.5 Provision the Newly Created Group

Now that the Greengrass group is created, it must be provisioned. The “Default Creation” option allows AWS IoT to create certificates, a core, and a role. The user selects the name of the group and the name of the core for the Greengrass group. Press the “Use default creation” button in the lower right-hand corner of the screen.

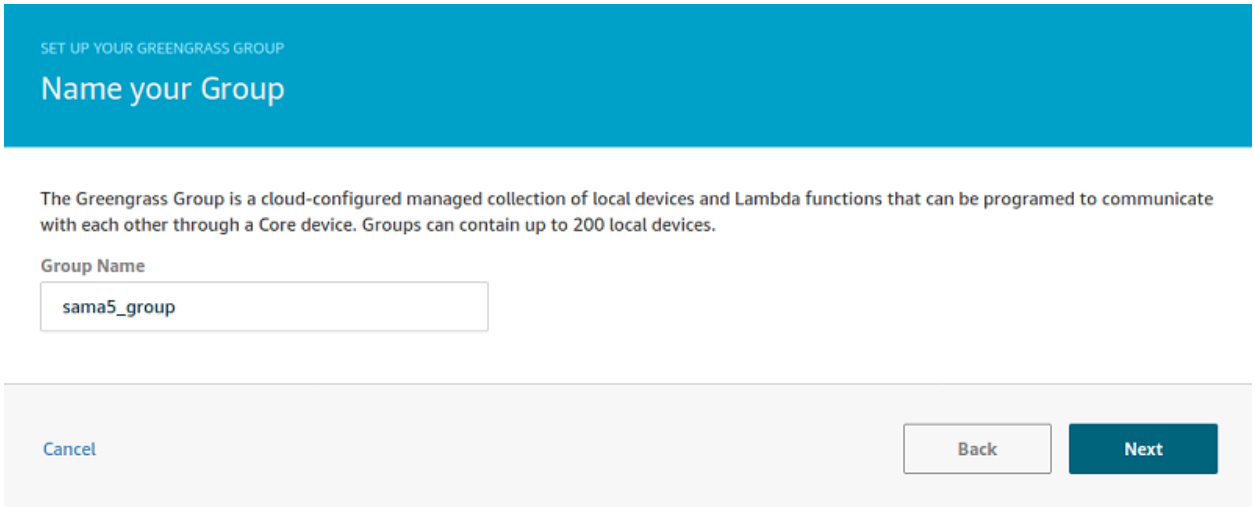


2.2.6 Name the Greengrass Group

The first step of the default creation process is to name the group.

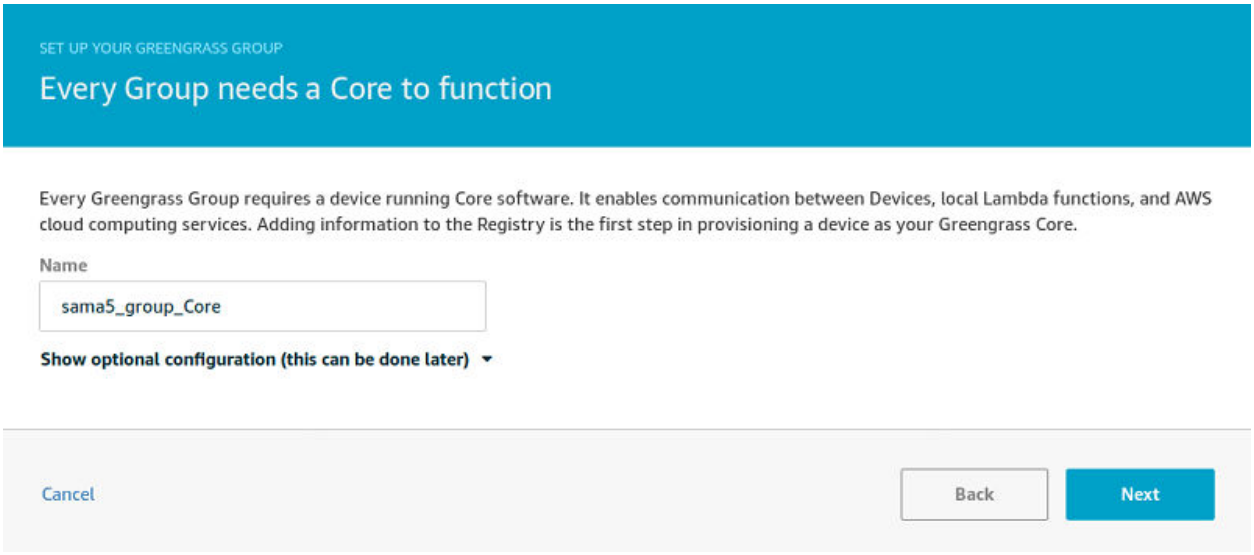
A Greengrass group is a representation of the Greengrass core (SAMA5D27-WLSOM1-EK running the Greengrass core software), local devices that communicate with the core, and Lambda functions that run on the core.

See the example below.



2.2.7 Name the Greengrass Core

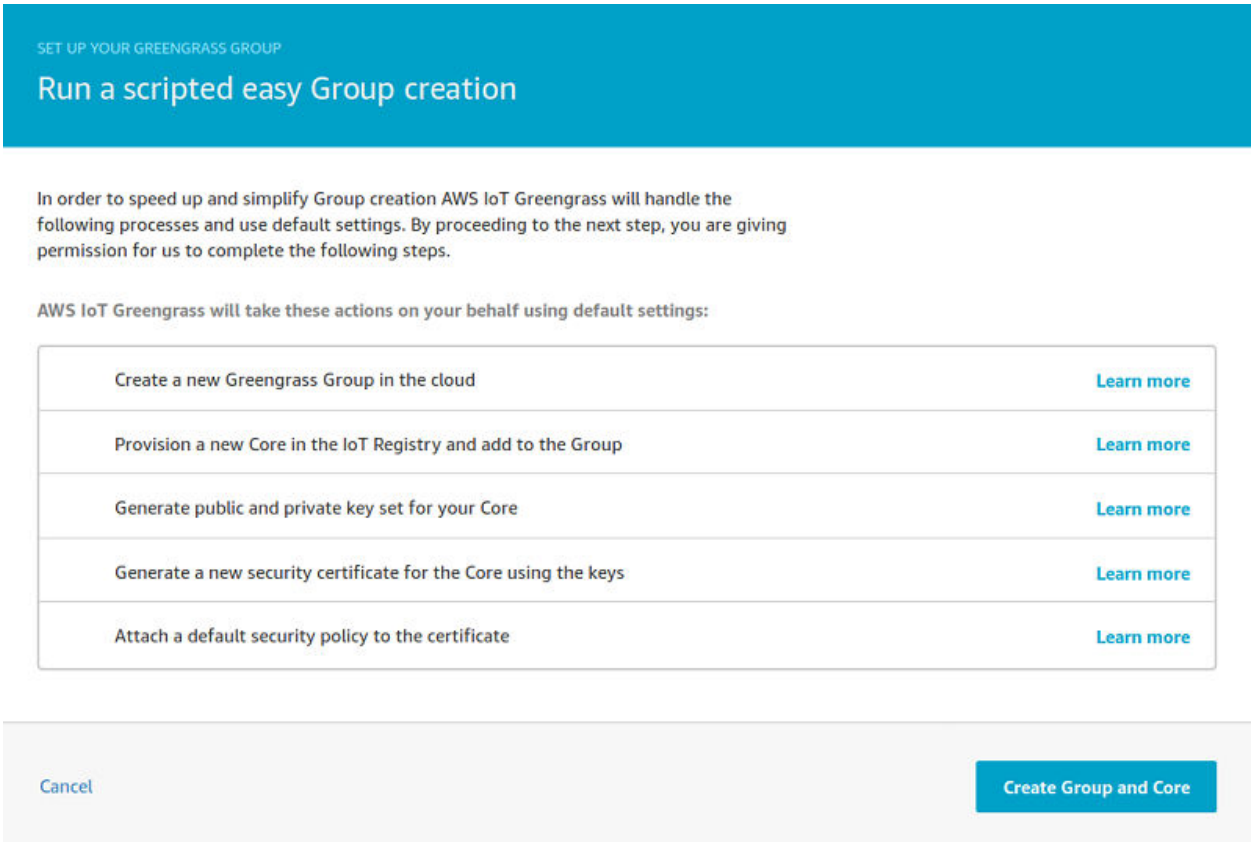
The Greengrass core is the SAMA5D2 system that runs the Greengrass core software. See the example name below.



2.2.8 Create the Group and Core

Since we are using the “Default Creation” wizard, AWS IoT is going to perform most of the work for provisioning. Press the “Create Group and Core” button to perform the following:

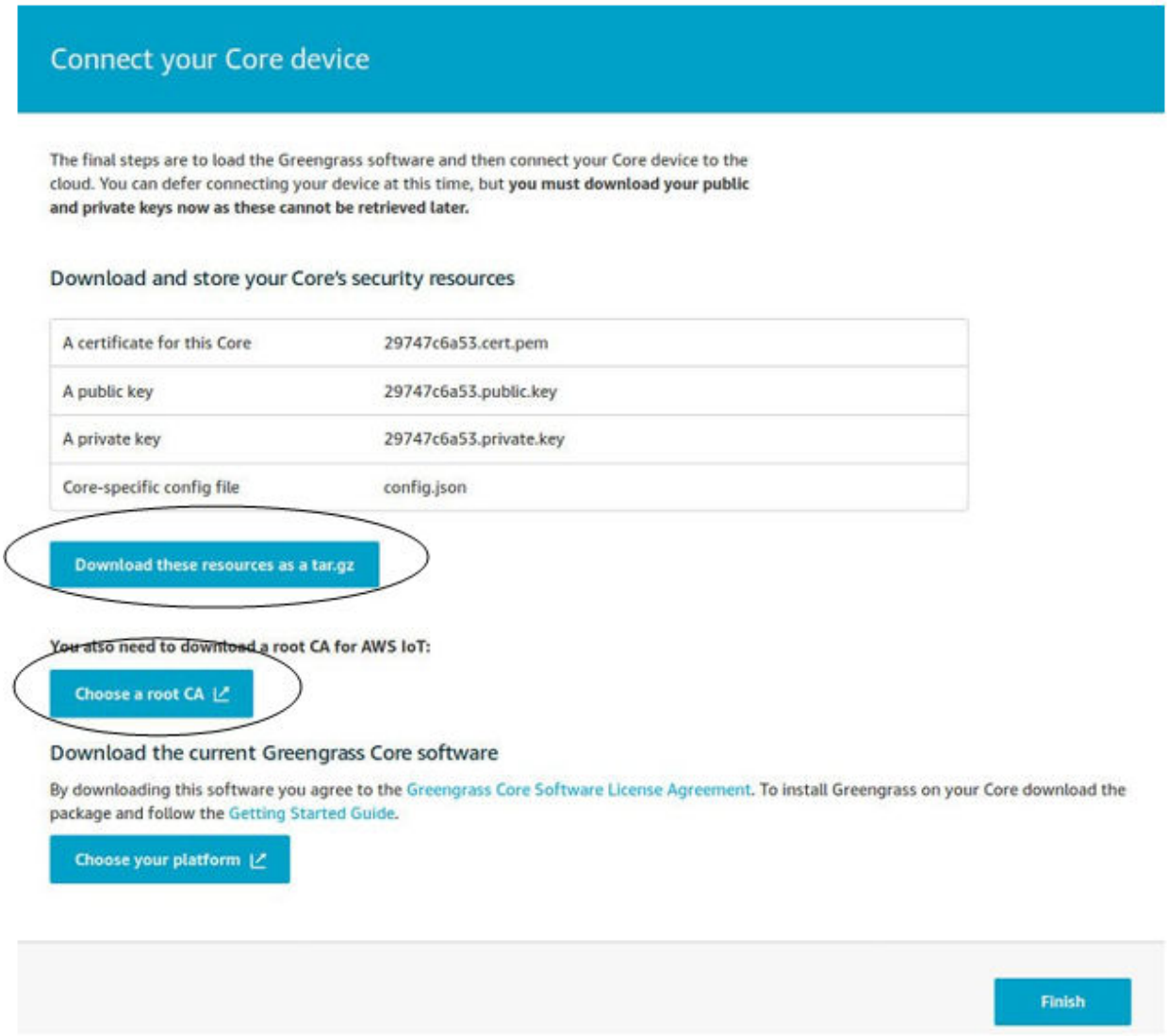
1. Create the Greengrass group.
2. Create a Greengrass core representing the SAMA5D2 system.
3. Create keys and certificates that will later be downloaded to the SAMA5D2.
4. Attach a security policy to the Greengrass core certificate.



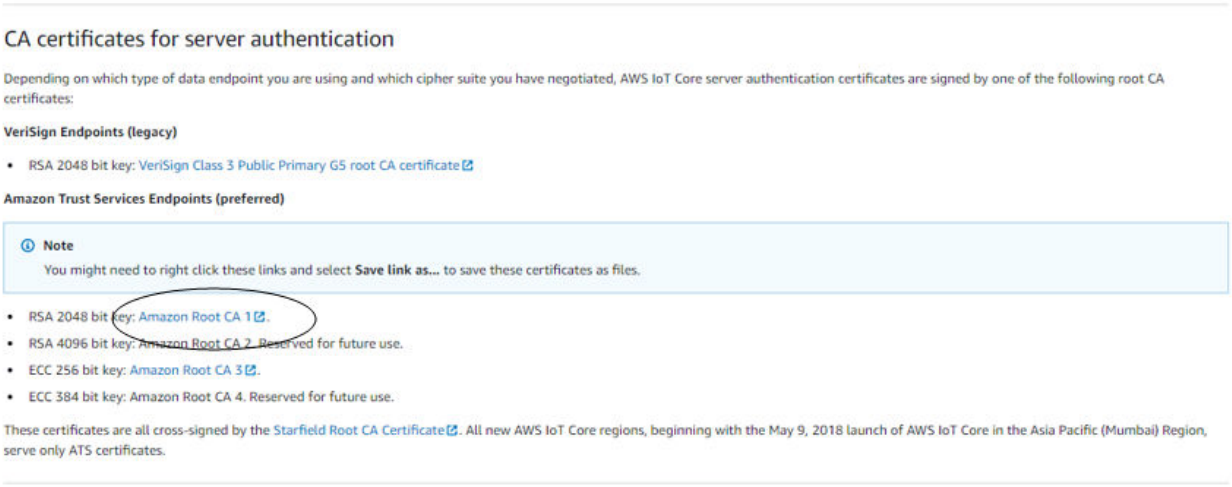
2.2.9 Download Certificates and Core Software

It is extremely important to download the keys and certificates for client authentication now. Although certificates can be downloaded at any time, this is the only opportunity to download the private key. When using Hardware Security Integration (HSI), these keys will be overwritten. Unless you are very comfortable with HSI, it is a good idea to try out Greengrass without HSI at first. After the Greengrass core is totally functional, then you can use the Hardware Secure Element to implement HSI to add another layer of security to the Greengrass core.

These downloaded credentials are used by the AWS endpoint to authenticate the Greengrass core. TLS Client authentication is the mechanism used for this validation. Make sure the download archive file is protected against unauthorized copying by storing in a secure location.

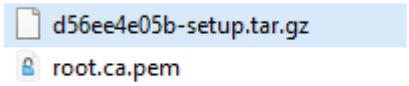


After downloading the keys and certificate, you must also download a Root CA certificate that corresponds to the AWS endpoint to which the Greengrass core talks. Press the “Choose a root CA” button and then download the correct certificate for your endpoint. In this example, the certificate is “Amazon Root CA 1”. This root CA certificate is the certificate that validates the AWS endpoint, and is not necessarily the same certificate used to sign the client certificate.



Rename “Amazon Root CA 1” to root.ca.pem.

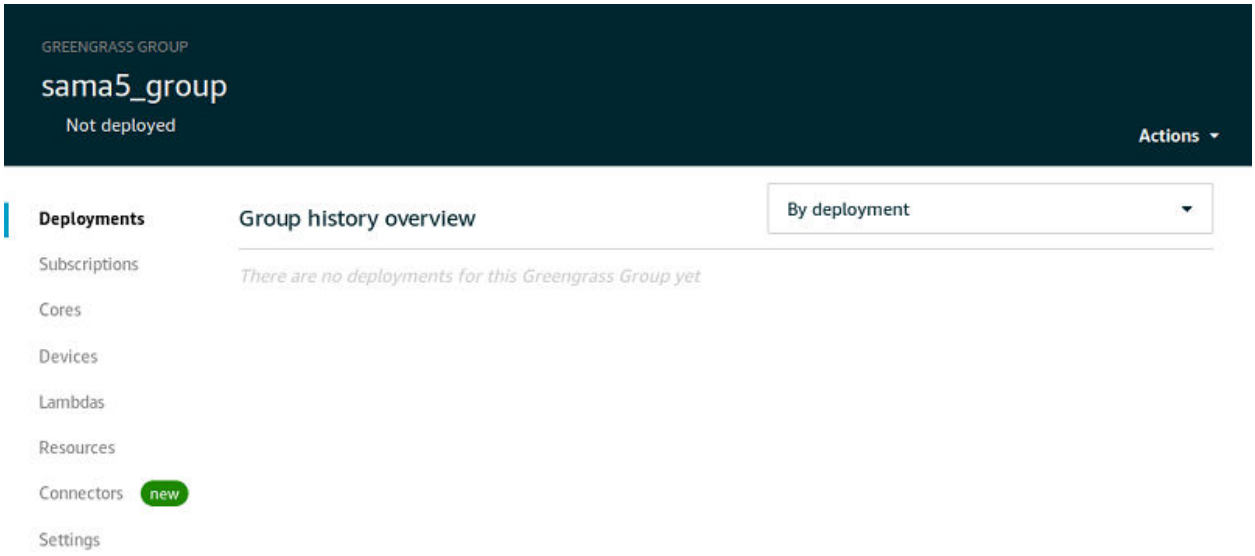
After downloading the resource as a targ.gz file, downloading “Amazon Root CA 1” and renaming it to root_ca_pem, files similar to the ones shown below should be displayed on your host PC:



AWS IoT Greengrass core software is already included in the Microchip Linux distribution.

3. Greengrass Group Creation Successful

At this stage, the Greengrass group is created.



4. Languages Used By Greengrass

Although many languages can be used to implement AWS Greengrass Lambda functions, this application note uses the Python® 3.7 programming language.

Other languages that can be used for Lambda functions are:

- Node.JS 6.10
- Java 8
- C
- C++
- Any language that supports importing C libraries

4.1 Building the Target Image

The software for the target system can be built using the Buildroot tool. This process is performed on a Linux host PC. Note that Greengrass core software is included in the Microchip Linux distribution.

Note: The steps below describe how to build an SD card image from scratch and are for information only. This procedure is not required.

"$" indicates a command on the host PC.

"#" indicates a command on the target board.

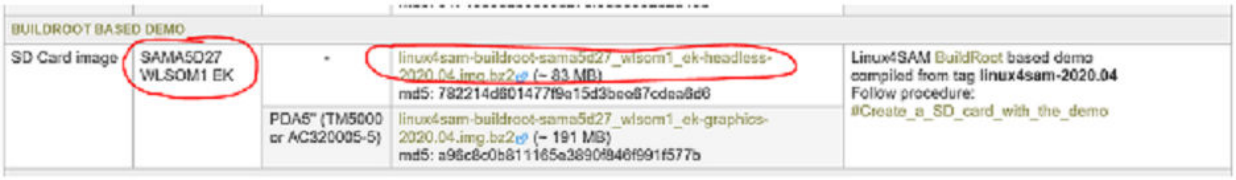
1. Change directory to a base location. In this example, cd to your home directory:
   $ cd
2. Create gg directory:
   $ mkdir gg
3. Change to the new directory:
   $ cd gg
4. Clone the following buildroot-external repository:
   $ git clone git://github.com/linux4sam/buildroot-external-microchip
5. Clone and checkout linux4sam buildroot repository:
   $ git clone git://github.com/linux4sam/buildroot-at91.git -b linux4sam_2020.04
6. Change to the buildroot directory:
   $ cd buildroot-at91/
7. Set up the configuration for buildroot:
   $BR2_EXTERNAL=../buildroot-external-microchip make sama5d27-wlsom1-ek_headless_defconfig
8. Make the buildroot project:
   $ make

At this point, Buildroot has created an entire SD card image "./output/images/sdcard.img". This image should now be copied to an SD card. You can use any number of different tools, such as Etcher, or you can carefully use the "dd" command that is built into Linux. Be careful that your destination drive is the SD card, because if you type the wrong name, a hard drive on your host PC could be erased.

4.2 Copy the Target Image to an SD Card

The SD card image available on the Linux4sam web site can be used directly, as the Greengrass core software is already part of the Microchip Linux distribution.

Choose the image for your target: www.linux4sam.org/bin/view/Linux4SAM/Sama5d27WLSom1EKMainPage.



You can use the image for other boards. For more information, refer to www.linux4sam.org/bin/view/Linux4SAM.

4.3 Copy Certificates and Root CA

Copy from the Host:

With the SD card mounted on the host filesystem, untar certificates to the SD card.

Assuming the SD card partition 2 is mounted on /mnt and certificates were downloaded to the user “Download” directory, perform the following:

```
$ cd /mnt
$ cd greengrass
$ sudo tar -xzvf ~/Downloads/<certprefix>-setup.tar.gz
$ sudo cp ~/Downloads/AmazonRootCA1.pem certs/root.ca.pem
```

Certificates and Root CA are now on the SD card. Place the SD card into the SAMA5D27-WLSOM1-EK board and power it.

Log-in is root.

An alternative means to copy Certificates and Root CA from the Host to the target is to use the USB stick.

- Plug the USB stick on the board.
- On the target, type the following commands:
  ```
  # mkdir media
  # mount /dev/sda /media/
  # cd /greengrass/certs
  # cp /media/* .
  # umount /dev/sda
  ```
- Unplug the USB stick.

As a result, all certificates including root.ca.pem should be on the target under greengrass/certs directory.

# indicates a command on the target.

```
# pwd
/greengrass/certs
# ls
d56ee4e05b.cert.pem     d56ee4e05b.public.key
d56ee4e05b.private.key  root.ca.pem
#
```

4.4 Edit Greengrass Configuration

```
# edit /greengrass/config/config.json
```

“useSystemd” must be set to “no”.

```
# pwd
/greengrass/config
# vi config.json
        "certPath": "[CLOUD_PEM_CRT_HERE]",
        "keyPath": "[CLOUD_PEM_KEY_HERE]",
        "thingArn": "[THING_ARN_HERE]",
        "iotHost": "[HOST_PREFIX_HERE]-ats.iot.[AWS_
        "ggHost": "greengrass-ats.iot.[AWS_REGION_HE
    },
    "runtime": {
        "cgroup": {
            "useSystemd": "no"
        }
    },
    "managedRespawn": false,
```

4.4.1 Edit Greengrass Configuration to Use Port 443 (optional)

The default configuration for Greengrass uses ports 8883 and 8443. In some environments, these ports may be blocked by firewalls. Greengrass can be configured to use port 443 instead. This is the same port used by the “https” protocol.

iotMqttPort=443 may be needed. MQTT communicates only via https-port=443 which is typically open in firewalls, as needed for HTTPS. However, its default-port 8883 is typically closed and IT usually does not open it.

This step is performed on the target of the SAMA5D27-WLSOM1-EK board.

Edit the file /greengrass/config/config.json to have the iotMqttPort, iotHttpPort, and ggHttpPort parameters as shown below:

```
# vim /greengrass/config/config.json
{
  "coreThing" : {
    "caPath" : "root.ca.pem",
    "certPath" : "2222222222.cert.pem",
    "keyPath" : "2222222222.private.key",
    "thingArn" : "arn:aws:iot:<region>:<account>:thing/sama5_group_Core",
    "iotHost" : "<endpoint>”,
    "iotMqttPort" : 443,
    "iotHttpPort" : 443,
    "ggHost" : "greengrass-ats.iot.<region>.amazonaws.com",
    "ggHttpPort" : 443,
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "no"
    }
  },
  "managedRespawn" : false,
  "crypto" : {
    "principals" : {
      "SecretsManager" : {
        "privateKeyPath" : "file:///greengrass/certs/2222222222.private.key"
      },
      "IoTCertificate" : {
        "privateKeyPath" : "file:///greengrass/certs/2222222222.private.key",
        "certificatePath" : "file:///greengrass/certs/2222222222.cert.pem"
      }
    },
    "caPath" : "file:///greengrass/certs/root.ca.pem"
  }
}
```

4.5 SAM5D27-WLSOM1-EK Setup for the Buildroot Image

4.5.1 edit /etc/wpa_supplicant.conf file

```
ctrl_interface=/var/run/wpa_supplicant
ap_scan=1

network={
  scan_ssid=1
  key_mgmt=WPA-PSK
  ssid="YourSSID"
  psk="YourPassword"
}
```

4.5.2 edit /etc/network/interfaces

```
# interface file auto-generated by buildroot
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
  pre-up /etc/network/nfs_check
  wait-delay 15
  hostname $(hostname)

auto wlan0
iface wlan0 inet dhcp
```

4.5.3 Wi-Fi Setup

Refer to the tutorials: https://www.linux4sam.org/bin/view/Linux4SAM/WilcFaq.

```
# modprobe wilc_sdio
# ifconfig wlan0 up
# wpa_supplicant -Dn180211 -iwlan0 -c/etc/wpa_supplicant.conf
# udhcpc -i wlan0
```

The device is set in Station mode.

Check the Wi-Fi connection:

```
#ping google.com
```

4.6 Add ggc_user and ggc_group

Greengrass core software assumes that ggc_user and ggc_group are on the Linux system. Add them as follows:

```
# adduser -S ggc_user
# addgroup -S ggc_group
```

Ensure that the date and time on your target are correct:

```
# date mmddhhmmyyyy
```

4.7 Start the Greengrass Core

On the SD card, a file named greengrass is found in the /etc/init directory. This start-up script starts the Greengrass software after performing housekeeping tasks. Start Greengrass with:

```
# /etc/init.d/greengrass start
```

The Greengrass system is now running.

Log file: runtime.log

Check/tail log file to see if there is any progress:

(target)# tail -F /greengrass/ggc/var/log/system/runtime.log

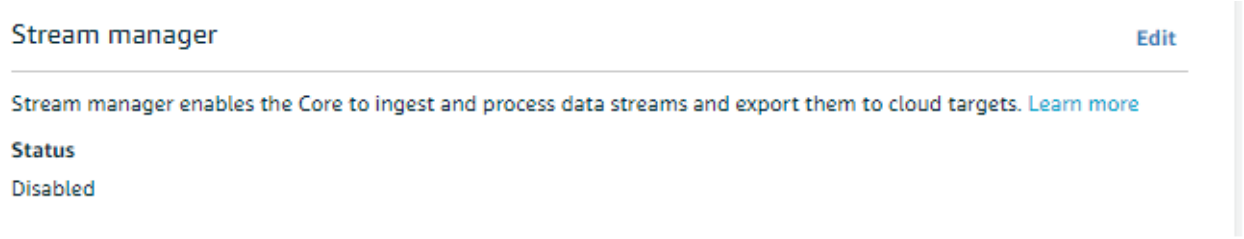
4.8 Deploy the Group from AWS Console

Select AWS IoT Greengrass, and select Groups. Then press the group that you created.

You can use a Wi-Fi connection for the host and the target.

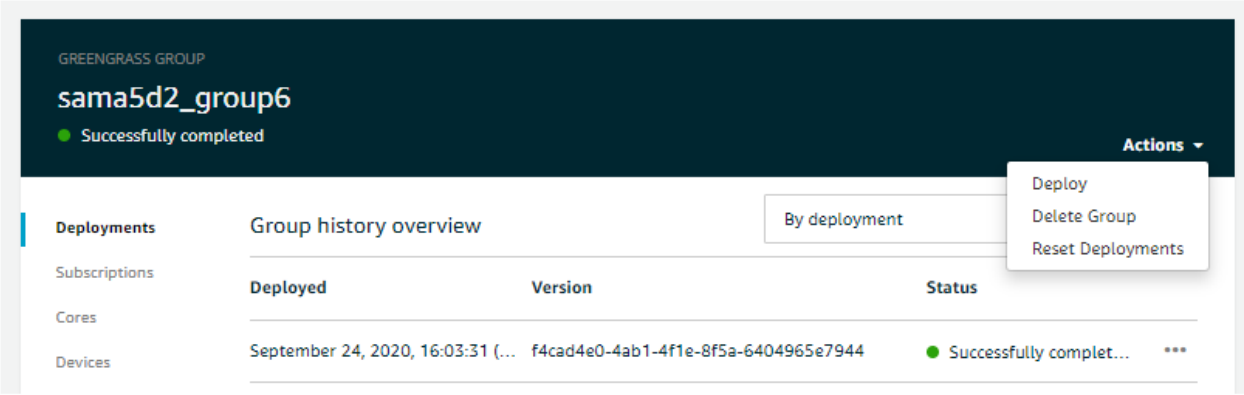
A mobile phone can be used as an access point to avoid potential problems with firewalls.

Go to the “Settings” section to disable the Stream manager feature.



To deploy the group, in the upper right of the screen, under the “Actions” menu, select “Deploy”.

This copies files from the AWS servers to the SAMA5D2 core system.



While deployment is ongoing, you should see traffic in the 'tail' logfile on the target. Once completed, the status in AWS changes to “Successfully completed”, as shown above.

5. Next Steps

You now have a running Greengrass system.

Modules 1 and 2 in the AWS tutorial do not apply to the SAMA5D27-WLSOM1-EK system. This application note has walked you through those steps. You can now follow tutorials starting with module 3 at the link below:

https://docs.aws.amazon.com/greengrass/latest/developerguide/module3-I.html

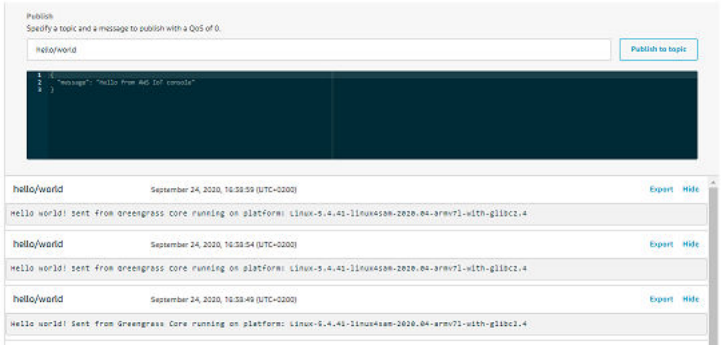
- Start the Lambda function as defined in the Module 3 (part1): "Create and package a Lambda function with Python3.7”
- In this step, you:
  - Download the AWS IoT Greengrass Core SDK for Python to your computer (not the AWS IoT Greengrass core device).
  - Create a Lambda function deployment package that contains the function code and dependencies.
    - Python 3.7 is not available on the Linux distribution which is required for the Lambda example, therefore you need to apply the following commands on the target:

```
# which python
/usr/bin/python
#cd /usr/bin/python
#ln -s python3.8 python3.7
```

  - Use the Lambda console to create a Lambda function and upload the deployment package.
  - Publish a version of the Lambda function and create an alias that points to the version.
  - To complete this module, Python 3.7 must be installed on your core device (as defined above).

As a result, the “hello world” message appears on the AWS console.

This message is sent from the SAM5D27-WLSOM1-EK Greengrass core to the AWS console (to the cloud).



6. Summary

Below is a summary of the steps to get started with AWS Greengrass, and to create a Lambda function:

1. Create a Greengrass group on the AWS console (need AWS account).
2. Download and extract the Greengrass core's security resources with certs/ and config/ directories.
3. Download the root CA for AWS IoT and paste it in certs/root.ca.pem.
4. Edit config/config.json and put "useSystemd" to "no".
5. Be careful regarding a firewall. A Mobile Access Point could be used for the host PC and the target.
6. Copy (overwrite) certs/ and config/ to target board in /greengrass directory of the target.
7. In the AWS console, disable Stream Manager for the group.
8. Start Greengrass by running "/etc/init.d/greengrass start".
9. Go to the AWS console and deploy the group. The deployment should have the status "Successfully completed".
10. Follow the guidelines and the different steps on this web site: https://docs.aws.amazon.com/greengrass/latest/developerguide/module3-l.html
    - Create and package a Lambda function.
    - Be careful to respect the zip hierarchy file.
    - Publish.
    - Create an alias.
    - Configure the Lambda function for AWS IoT Greengrass.
    - Deploy cloud configurations to a core device.
    - Verify the Lambda function is running on the core device.

7. Greengrass System Requirements

The AWS Greengrass documentation describes several requirements of the Linux system. The Buildroot system makes sure the following items are enabled.

The following items are required:

- Minimum 128 MB RAM allocated to the AWS IoT Greengrass core device.
- Linux kernel version 4.4 or greater:
- Glibc library version 2.14 or greater.
- The /var/run directory must be present on the device.
- Hardlink and symlink protection
- The following Linux kernel configurations must be enabled on the device:
  - Namespace: CONFIG_IPC_NS, CONFIG_UTS_NS, CONFIG_USER_NS, CONFIG_PID_NS
  - CGroups: CONFIG_CGROUP_DEVICE, CONFIG_CGROUPS, CONFIG_MEMCG
  - Others: CONFIG_POSIX_MQUEUE, CONFIG_OVERLAY_FS, CONFIG_HAVE_ARCH_SECCOMP_FILTER, CONFIG_SECCOMP_FILTER, CONFIG_KEYS, CONFIG_SECCOMP
- dev/stdin, /dev/stdout, and /dev/stderr must be enabled
- The Linux kernel must support cgroups in order to run AWS IoT Greengrass with containers.
- The memory cgroup must be enabled and mounted to allow AWS IoT Greengrass to set the memory limit for Lambda functions.
- The root certificate for Amazon S3 and AWS IoT must be present in the system trust store.

The following items are optional:

- The devices cgroup must be enabled and mounted if Lambda functions with Local Resource Access (LRA) are used to open files on the AWS IoT Greengrass core device.
- Python version 3.7 is required if Python Lambda functions are used. If so, ensure that it is added to your PATH environment variable.
- The following commands are required for Greengrass OTA Agent: wget, realpath, tar, readlink, basename, dirname, pidof, df, grep, and umount.

8. Secure Element

After successfully getting Greengrass running using downloaded credentials from AWS, you can now implement Greengrass HSI using the ATECC608A Secure Element.

These steps are performed on the SAMA5D27-WLSOM1-EK board.

8.1 Configuring cryptoauthlib PKCS11 Library

By default, the following files are created:

- /etc/cryptoauthlib/cryptoauthlib.conf
```
# Cryptoauthlib Configuration File
filestore = /var/lib/cryptoauthlib
```
- /var/lib/cryptoauthlib/slot.conf.tmpl
```
# Reserved Configuration for a device
# The objects in this file will be created and marked as undeletable
# These are processed in order. Configuration parameters must be comma
# delimited and may not contain spaces

interface = i2c,0xB0
freeslots = 1,2,3

# Slot 0 is the primary private key
object = private,device,0

# Slot 10 is the certificate data for the device's public key
#object = certificate,device,10

# Slot 12 is the intermediate/signer certificate data
#object = certificate,signer,12

# Slot 15 is a public key
object = public,root,15
```

8.1.1 cryptoauthlib.conf

This file provides the basic configuation information for the library. The only variable is “filestore” which is where cryptoauthlib will find device specific configuration and where it will store object files from pkcs11 operations.

8.1.2 slot.conf.tmpl

This is a template for device configuration files that cryptoauthlib uses to map devices and their resources into pkcs11 tokens and objects.

A device file must be named <pkcs11_slot_number>.conf

For a single device:

cd /var/lib/cryptoauthlib

cp slot.conf.tmpl 0.conf

Then edit 0.conf to match the device configuration being used. In this case, change the interface line from

```
interface = i2c,0xB0
```

to

```
interface = i2c,0xC0
```

8.1.3 interface

Allows values: 'hid', 'i2c'

If using i2c specify the address in hex for the device. This is in the device format (upper 7 bits define the address) so will not appear the same as the i2cdetect address (lower 7 bits).

8.1.4 freeslots

This is a list of slots that may be used by the library when a pkcs11 operation that creates new objects is used. When the library is initialized, it scans for files of the form <pkcs11_slot_num>.<device_slot_num>.conf, which defines the object using that device resource.

8.2 Using p11-kit-proxy

1. Create or edit the global configuration file: /etc/pkcs11/pkcs11.conf

```
# This setting controls whether to load user configuration from the
# ~/.config/pkcs11 directory. Possible values:
#    none: No user configuration
#    merge: Merge the user config over the system configuration (default)
#    only: Only user configuration, ignore system configuration
user-config: merge
```

2. Create a module configuration file: **/usr/share/p11-kit/modules/cryptoauthlib.module**

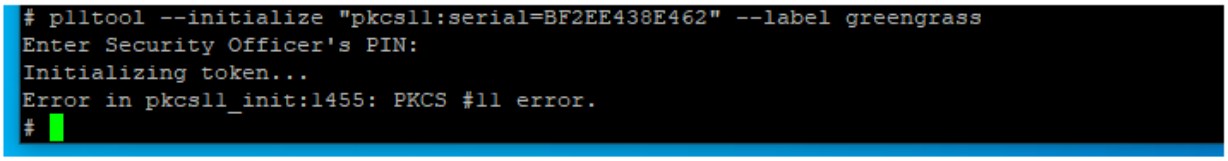
```
module: /usr/lib/libcryptoauth.so
critical: yes
trust-policy: yes
managed: yes
log-calls: no
```

For more details on the configuration files, see the configuration documentation.

8.3 Device Initialization Using P11tool

To initialize the device with a basic configuration (known as the standard TLS configuration) using p11tool:

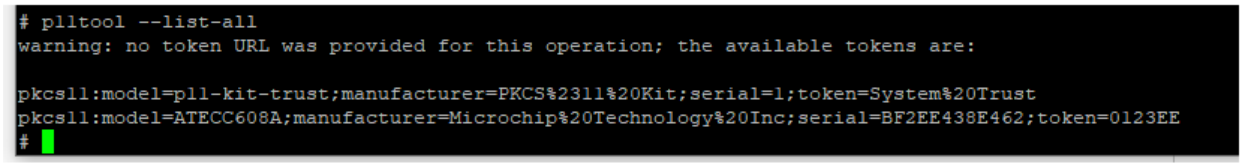
```
# p11tool --initialize "pkcs11:serial=BF2EE438E462" --label greengrass
Enter Security Officer's PIN:123
Initializing token... done
```




```
Token was successfully initialized; use --initialize-pin and --initialize-so-pin to
set or reset PINs
```

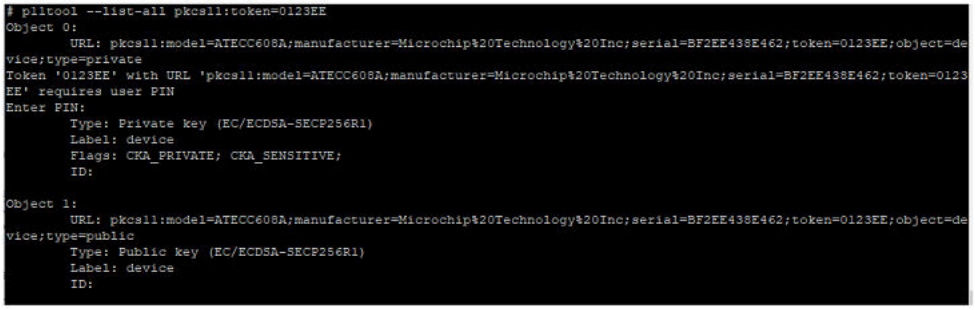
The error in pkcs11_init can be ignored.

The device must be identified in some way to p11tool using the pkcs11 string. In this example, the serial number previously obtained from the `p11tool --list-all` command is used. The label is a required field but is currently treated as a dummy value, as the library provides the value and it will be a field in the configuration file in the future.



8.4 Verifying the Initialization

Once the initialization/configuration is complete, rerunning the p11tool --list-all command displays the required objects:



At this point, all the tests listed at the end of the pkcs11 readme can be conducted. However, these tests are unnecessary as we move through the next steps for configuring Greengrass.

8.5 Probing the device

An uninitialized device with the defaults provided in the readme displays the following:

```
# p11tool –list-all pkcs11:token=0123EE
Object 0:
        URL: pkcs11:model=ATECC608A;manufacturer=Microchip%20Technology
%20Inc;serial=9F9CB19FF7BF;token=0123EE;object=device;type=private
        Type: Private key
        Label: device
        Flags: CKA_PRIVATE; CKA_SENSITIVE;
        ID:
```

8.5.1 Troubleshooting

If the device does not appear at all:

```
# p11tool –list-all pkcs11:token=0123EE
p11-kit: ateccx08: module failed to initialize: An error occurred on the device
pkcs11_init: PKCS #11 initialization error.
warning: no token URL was provided for this operation; the available tokens are:
```

Probe the bus and obtain the actual device address:

```
# i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Remember the expected format for the device address is shifted left 1 bit from the value returned from i2cdetect. Thus edit /var/lib/cryptoauthlib/0.conf with the probed value (0x60 becomes 0xC0 when shifted):

```
# Reserved Configuration for a device
# The objects in this file will be created and marked as undeletable
# These are processed in order. Configuration parameters must be comma
# delimited and may not contain spaces

interface = i2c,0xC0
freeslots = 1,2,3

# Slot 0 is the primary private key
object = private,device,0

# Slot 10 is the certificate data for the device's public key
#object = certificate,device,10
```

```
# Slot 12 is the intermedate/signer certificate data
#object = certificate,signer,12

# Slot 15 is a public key
object = public,root,15
```

To initialize the device with a basic configuration (known as the standard TLS configuration) using p11tool:

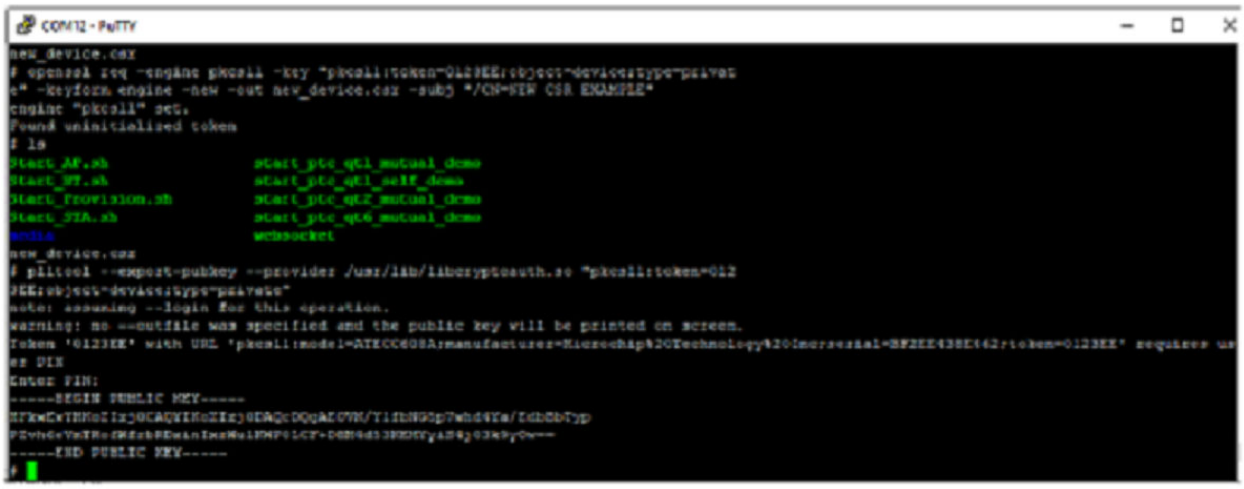
```
# p11tool --initialize "pkcs11:serial=BF2EE438E462" --label greengrass
Enter Security Officer's PIN:
Initializing token...
done

Token was successfully initialized; use --initialize-pin and --initialize-so-pin to
set or reset PINs
```

8.6 Setting Up the Greengrass Certificate

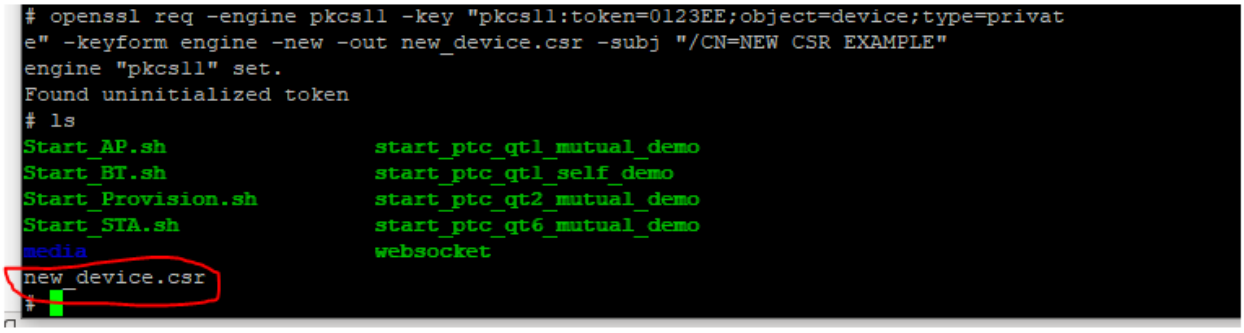
The public key from the ECC608 device:

```
# p11tool --export-pubkey --provider /usr/lib/libcryptoauth.so
"pkcs11:token=0123EE;object=device;type=private"
```



The Greengrass instance should already be set up. In order to use the hardware keys rather than the AWS provided keys, generate a Certificate Signed Request (CSR) using openssl:

```
 #openssl req -engine pkcs11 -key "pkcs11:token=0123EE;object=device;type=private" -
keyform engine -new -out new_device.csr -subj "/CN=NEW CSR EXAMPLE
```



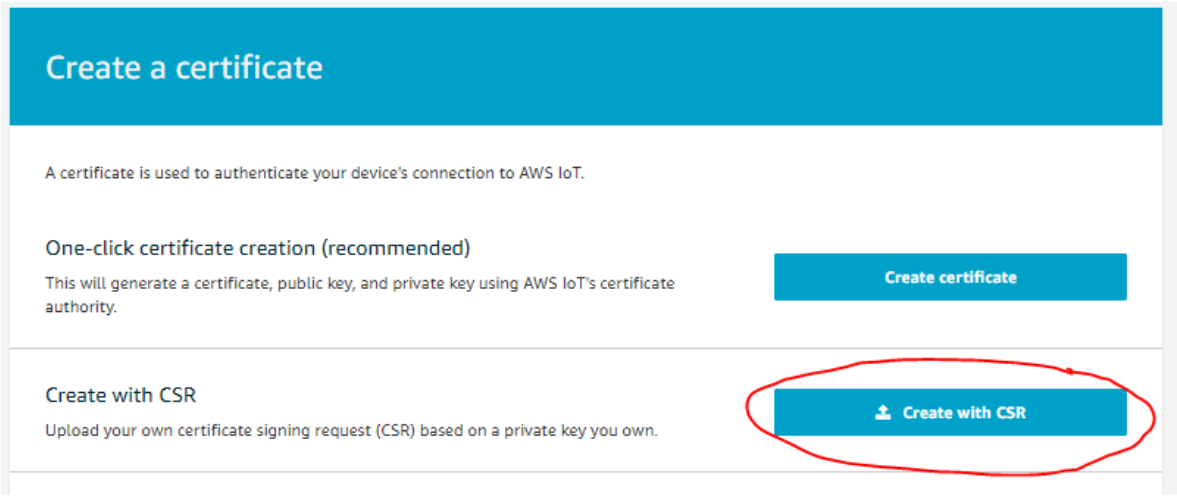
To verify the CSR was created correctly:

```
# openssl req -in new_device.csr -verify -text -noout
verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: CN = NEW CSR EXAMPLE
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:39:52:bf:63:57:db:34:6e:69:ef:08:5d:e1:86:
                    bf:11:d6:c1:6d:3c:a9:3d:9b:e1:e9:e5:66:4d:1a:
                    1f:59:fa:db:44:3c:e2:9c:8c:6b:5a:e9:4a:58:fd:
                    0b:08:5f:83:04:ce:1d:e7:72:84:31:8c:a2:4b:88:
                    f4:de:4f:72:3b
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        Attributes:
            a0:00
    Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:f1:71:91:ac:db:62:64:a3:aa:04:12:ec:48:
         38:94:7d:39:d8:15:f7:6a:a6:7b:5f:22:74:8f:67:7d:06:0c:
         3f:02:21:00:ed:32:59:c6:17:4b:89:68:8f:30:06:8c:c5:4e:
         e0:d4:c9:64:e7:93:03:87:43:1c:f9:e4:ab:59:8e:bc:7c:f9
#
```

8.6.1 Submit the CSR to AWS to Obtain the Connection Certificate

To obtain the connection certificate, use the AWS console:

1. Browse to Greengrass-> Manage -> Things -> Security.
2. Click the “View other options” button. This provides a menu of options.
3. To use the CSR generated, click the “Create with CSR” button and provide the new_device.csr file.
4. Click the “Upload CSR” button. This should give a “Certificate Created!” success screen.
   – Copy the generated CSR file from the target to the host.
   – Upload your CSR file from the host to the AWS console.



5. Upload and download the certificate provided and save it to `/greengrass/certs/` on the SAMA5D2 platform.

6. Before closing the screen, be sure to click the “Activate” the certificate to allow connections to AWS.
7. Click the “Attach a policy” button and attach the Greengrass core policy created during the Greengrass tutorial

8.6.2 Edit the config.json file to Use the pkcs11 Provider

This section duplicates the information provided in AWS documentation.

The final step is to modify the `/greengrass/config/config.json` file to inform Greengrass of the pkcs11 provider.

First, remove the `caPath, certPath`, and `keyPath` properties from the `coreThing` object.

```
{
 "coreThing" : {
 "caPath": "root-ca-pem",
 "certPath": "cloud-pem-crt",
 "keyPath": "cloud-pem-key",
 ...
 },
 ...
}
```

If using p11-kit:

```
{
  "crypto": {
          "caPath": "file:///greengrass/certs/root.ca.pem",
          "PKCS11": {
                  "OpenSSLEngine": "/usr/lib/engines-1.1/pkcs11.so",
                  "P11Provider": "/usr/lib/p11-kit-proxy.so",
                  "slotLabel": "0123301",
                  "slotUserPin":
"00112233445566778899AABBCCDDEEFF00112233445566778899AABBCCDDEEFF"
          },
          "principals": {
                  "IoTCertificate": {
                          "privateKeyPath":
"pkcs11:token=012301;object=device;type=private",
                          "certificatePath": "file:///greengrass/certs/1cc2e5fa99-
certificate.pem.crt"
                  }
          }
  },
  "coreThing" : {
    "thingArn" : "arn:aws:iot:eu-central-1:96949751109:thing/sam5d2_group6_Corre"
    "iotHost" :  "a2lp13dce8v5g3-ats.iot.eu-central-1.amazonaws.com",
    "ggHost" : "greengrass-ats.iot.eu-central-1.amazonaws.com",
    "keepAlive" : 600
  },
  "runtime" : {
    "cgroup" : {
      "useSystemd" : "no"
    }
  },
  "managedRespawn" : false
}
```

8.7 Summary

By following the procedures detailed in this application note, you should now be able to implement Cloud Edge Services using Microchip MPUs and AWS IoT Greengrass.

8.8 Additional Resources

Microchip MPUs – http://www.microchip.com/mpu

Amazon Web Services – http://aws.amazon.com

AWS Management Console – https://console.aws.amazon.com

AWS IoT Greengrass Developer guide - https://docs.aws.amazon.com/greengrass/latest/developerguide

9. Revision History

9.1 Rev. B - 10/2020

Screenshots added and modified throughout. Code snippets modified throughout.
Prerequisites: modified development board reference.
Setting Up the Hardware: modified.
Provision the Newly Created Group: changed to Default creation option.
Download Certificates and Core Software: updated.
Languages Used by Greengrass: modified Python version to 3.7.
Building the Target Image: updated.
Next Steps: new content added.
Summary: new section added.
Setting Up the Greengrass Certificate: updated.

9.2 Rev. A - 07/2019

First issue.

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-7118-9

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820