

Renesas RA Family

RA AWS MQTT/TLS Cloud Connectivity Solution

Introduction

This application note describes IoT Cloud connectivity solution in general, provides a brief introduction to IoT Cloud providers like Amazon Web Services (AWS), and covers the FSP MQTT/TLS module and its features. The application example provided in the package uses AWS IoT Core. The detailed steps in this document show first-time AWS IoT Core users how to configure the AWS IoT Core platform to run this application example.

This application note enables developers to effectively use the FSP MQTT/TLS modules in end-product design. Upon completion of this guide, developers will be able to add the AWS MQTT Client, Med TLS, Secure sockets on WiFi using Silex WiFi modules, configure them correctly for the target application, and write code using the included application example code as a reference and efficient starting point.

References to detailed API descriptions, and other application projects that demonstrate more advanced uses of the module, are in the *FSP User's Manual*, which serves as a valuable resource in creating more complex designs.

This MQTT/TLS AWS Cloud Connectivity solution is supported on EK-RA6M3 and EK-RA6M3G kits.

Required Resources

To build and run the MQTT/TLS application example, the following resources are needed:

Development tools and software

- e² studio ISDE v7.8.0 or later
- Flexible Software Package (FSP) 1.1.0 or later
- SEGGER J-link® USB driver ([renesas.com/synergy/jlinksynergy](https://www.renesas.com/synergy/jlinksynergy)).
- SEGGER RTT Viewer (segger.com/products/debug-probes/j-link/tools/rtt-viewer/)
- SEGGER SWO Viewer (segger.com/products/debug-probes/j-link/tools/j-link-swo-viewer/)

Hardware

- Renesas RA™ EK-RA6M3 kit (renesas.com/ra/ek-ra6m3).
- Renesas Silex PMOD based Wi-Fi Module (renesas.com/wi-fi-pmod).
- PC running Windows® 7 or 10; and an installed web browser (Google Chrome, Internet Explorer, Microsoft Edge, Mozilla Firefox, or Safari).
- Micro USB cables

Prerequisites and Intended Audience

This application note assumes that the user is adept in operating the Renesas e² studio ISDE with Flexible Software Package (FSP). If not, we recommend that they read and follow procedures in the *FSP User Manual* (FSPUM) sections for 'Starting Development' including 'Debug the Blinky Project'. Doing so enables familiarization with e² studio and FSP and validates proper debug connection to the target board. In addition, this application note assumes prior knowledge of MQTT/TLS and its communication protocols.

The intended audience is users who want to develop applications with MQTT/TLS modules using Renesas RA™ RA6 MCU Series.

Prerequisites

1. Access to online documentation available in the Cloud Connectivity References section.
2. Access to latest documentation for identified Renesas Flexible Software Package.
3. Prior knowledge of operating e² studio and built-in (or standalone) RA Configurator.
4. Access to associated hardware documentation such as User Manuals, Schematics, and so forth.

Contents

1.	Introduction to Components for Cloud Connectivity	4
1.1	General Overview	4
1.2	Cloud Service Provider	4
1.3	AWS IoT Core	5
1.4	MQTT Protocol Overview	5
1.5	TLS Protocol Overview	5
1.6	Device Certificates, CA, and Keys	6
2.	AWS MQTT Client with RA FSP	6
3.	Secure Sockets Implementation	8
4.	Mbed TLS	9
5.	MQTT Module APIs Usage	10
6.	Cloud Connectivity Application Example	10
6.1	Overview	10
6.2	MQTT/TLS Application SW Architecture Overview	12
6.3	Creating the Application Project using the FSP Configurator	13
6.4	MQTT/TLS Configuration	18
6.4.1	IoT Cloud Configuration (AWS)	20
6.4.1.1	AWS IoT Policies	20
6.4.2	Creating a Device on AWS IoT Core	20
6.4.2.1	Open AWS IoT Core Service	20
6.4.2.2	Create a Thing	21
6.4.3	Generating Device Certificate and Keys	26
6.4.4	Creating a Policy for a Device	29
6.4.5	Connecting the Certificate to the Policy	30
6.5	Running the MQTT/TLS Application Example	31
6.5.1	Importing, Building and Loading the Project	31
6.5.1.1	Importing	31
6.5.1.2	Building the Latest Executable Binary	31
6.5.2	Loading the Executable Binary into the Target MCU	31
6.5.2.1	Using a Debugging Interface with e ² studio	31
6.5.2.2	Using J-Link Tools	31
6.5.2.3	Using Renesas Flash Programmer	32
6.5.3	Powering up the Board	32
6.5.3.1	Deviation from Default Jumper Settings	32
6.5.3.2	Additional Components to Connect	32
6.5.3.3	Power-on Behavior	32
6.6	Connecting to AWS IoT	32

6.6.1	WiFi Credentials	32
6.6.2	AWS IoT Credentials	33
6.7	Verifying the Application Project.....	34
7.	MQTT/TLS Module Next Steps.....	35
8.	Cloud Connectivity References.....	35
	Revision History	38
1.	Introduction to Components for Cloud Connectivity	4
1.1	General Overview	4
1.2	Cloud Service Provider.....	4
1.3	AWS IoT Core	5
1.4	MQTT Protocol Overview	5
1.5	TLS Protocol Overview.....	5
1.6	Device Certificates, CA, and Keys	6
2.	AWS MQTT Client with RA FSP	6
3.	Secure Sockets Implementation	8
4.	Mbed TLS.....	9
5.	MQTT Module APIs Usage	10
6.	Cloud Connectivity Application Example.....	10
6.1	Overview.....	10
6.2	MQTT/TLS Application SW Architecture Overview.....	12
6.3	Creating the Application Project using the FSP Configurator	13
6.4	MQTT/TLS Configuration	18
6.5	Running the MQTT/TLS Application Example	31
6.6	Connecting to AWS IoT	32
6.7	Verifying the Application Project.....	34
7.	MQTT/TLS Module Next Steps.....	35
8.	Cloud Connectivity References.....	35
	Revision History	38

1. Introduction to Components for Cloud Connectivity

1.1 General Overview

The Internet-of-Things (IoT) is a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. The 'things' in this definition are objects in the physical world (physical objects) or information world (virtual) that can be identified and integrated into communication networks. In the context of the IoT, a 'device' is a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage and data processing [1]. Communication is often performed with providers of network-hosted services, infrastructure, and business applications to process/analyze the generated data and manage the devices. Such providers are called Cloud Service Providers. While there are many manufacturers for devices and cloud service providers, for the context of this application note, the device is a Renesas RA Microcontroller (MCU) connecting to services provided by Amazon Web Services (AWS) for IoT.

1.2 Cloud Service Provider

AWS IoT is a platform that enables users to connect devices to AWS Services and other devices, secure data and interactions, process and act upon device data, and enable applications to interact with devices even when they are offline. As a Cloud Service Provider, AWS IoT provides the ability to:

1. Connect and manage devices
2. Secure device connections and data
3. Process and act upon device data
4. Read and set device state at any time

The following figure summarizes the features provided by AWS IoT [2]:

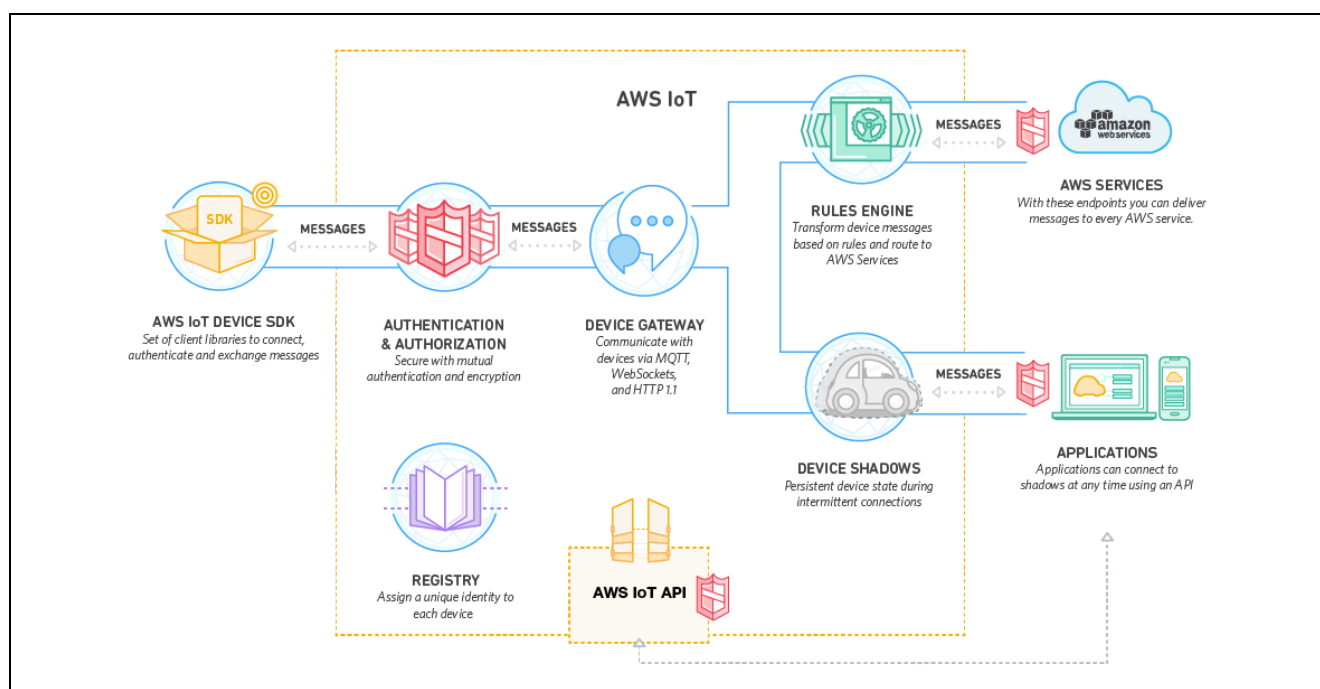


Figure 1. AWS IoT Features, Service Components, and Data Flow Diagram

A key feature provided by AWS is the AWS IoT Software Development Kit (SDK) written in C language to allow devices such as sensors, actuators, embedded micro-controllers, or smart appliances; to connect, authenticate, and exchange messages with AWS IoT using the MQTT, HTTP, or WebSocket's protocols. *This application note focuses on configuring and using the AWS IoT Device SDK and the included MQTT protocol available through the Renesas Flexible Software Package (FSP) for Renesas RA MCUs.*

1.3 AWS IoT Core

AWS IoT Core is a managed cloud service that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT Core can support billions of devices and trillions of messages. It can process and route messages to AWS endpoints and to other devices reliably and securely. With AWS IoT Core, customer applications can keep track of and communicate with all devices, all the time, even when they are not connected [3].

AWS IoT Core addresses security concerns for the infrastructure by implementing mutual authentication and encryption. AWS IoT Core provides automated configuration and authentication upon a device's first connection to AWS IoT Core, as well as end-to-end encryption throughout all points of connection, so that data is never exchanged between devices and AWS IoT Core without proven identity [3].

This application note focuses on complementing the security needs of AWS IoT Core through installing a proven identity for the RA MCU by storing a X.509 certificate and asymmetric cryptography keys in Privacy Enhanced Mail (PEM) format in the on-board flash. The RA MCU has on-chip security features, such as Key Wrapping, to protect the private key associated with the public key and the certificate associated with the device¹. Additionally, RA MCUs can also generate asymmetric keys using features of the Secure Cryptography Engine (SCE) and API available through the FSP. The SCE is used to accelerate symmetric encryption/decryption of data between the connected device and AWS IoT, allowing the ARM Cortex-M processor to perform other application specific computations.

1.4 MQTT Protocol Overview

Message Queuing Telemetry Transport (MQTT) is featured in this application note as it is a lightweight communication protocol specifically designed to tolerate intermittent connections, minimize the code footprint on devices, and reduce network bandwidth requirements. MQTT uses a publish/subscribe architecture which is designed to be open and easy to implement, with up to thousands of remote clients capable of being supported by a single server. These characteristics make MQTT ideal for use in constrained environments where network bandwidth is low or where there is high latency and with remote devices that might have limited processing capabilities and memory [4]. *The RA MCU device in this application note implements a MQTT Client which communicates with AWS IoT and exchanges example telemetry information, such as MCU temperature, and MCU GPIO status.*

1.5 TLS Protocol Overview

The primary goal of the Transport Layer Security (TLS) protocol is to provide privacy and data integrity between two communicating applications [5] or endpoints. AWS IoT mandates use of secure communication. Consequentially, all traffic to and from AWS IoT is sent securely using TLS [6]. TLS protocol version 1.2 is used to ensure the confidentiality of the application protocols supported by AWS IoT. A variety of TLS Cipher Suites are supported [7]. This application note configures the RA Flexible Software Package for the MCU based device to provide the following capabilities and AWS IoT negotiates the appropriate TLS Cipher Suite configuration to maximize security.

Table 1. TLS Capabilities in RA FSP

Secure Crypto Hardware Acceleration	Supported
Key Format Supported	AES, ECC, RSA
Hash	MD5, SHA-256
Cipher	AES
Public Key Cryptography	ECC, ECDSA, RSA
Message Authentication Code (MAC)	HKDF

On top of these above supported features, Mbed Crypto middleware also supports a variety of features which can be enabled through the RA Configurator. Refer to the FSP UM section for the Crypto Middleware (rm_psa_crypto).

¹ This application note does not focus on using Key Wrapping for securely storing the private key for devices deployed in a production environment.

1.6 Device Certificates, CA, and Keys

Device certificates, Certificate Authorities, and Asymmetric Key Pairs create the foundation for trust needed for a secure environment. The background information on these commonly used components in AWS is as follows:

A *digital certificate* is a document in a known format that provides information about the identity of a device. X.509 is a standard that includes the format definition for public-key certificate, attribute certificate, certificate revocation list (CRL) and attribute certificate revocation list (ACRL) [8]. X.509 defined certificate formats (X.509 Certificate) are commonly used on the internet and in AWS IoT for authenticating a remote entity/endpoint, that is, a Client and/or Server. In this application note, an X.509 certificate and asymmetric cryptography key pair (public and private keys) are generated from AWS IoT and installed (during binary compilation) into RA MCU device running the MQTT Client to establish a *known identity*. In addition, a root Certification Authority (CA) certificate is also downloaded and used by the device to authenticate the connection to the AWS IoT gateway.

Certification authority (CA) certificates are certificates that are issued by a CA to itself or to a second CA for the purpose of creating a defined relationship between the two CAs². The root CA certificate allows devices to verify that they are communicating with AWS IoT Core and not another server impersonating AWS IoT Core.

The public and private keys downloaded from AWS IoT use RSA algorithms for encryption, decryption, signing and verification³. These key pairs, and certificates are used together in the TLS process to:

1. Verify device identity.
2. Exchange symmetric keys, for algorithms such as AES, for encrypting and decrypting data transfers between endpoints.

2. AWS MQTT Client with RA FSP

The AWS MQTT library included in RA FSP can connect to either AWS MQTT or to any third party MQTT broker such as Mosquitto [9]. The complete documentation for the library can be found on the AWS IoT Device SDK C: MQTT website [10]. Primary features supported by the library are:

- MQTT connections over TLS to an AWS IoT Endpoint or Mosquitto server or any MQTT broker.
- Non-secure MQTT connections to Mosquitto servers.⁴

The AWS MQTT Client can be directly imported into a Thread Stack and is configured through the RA Configuration Perspective. To add the AWS MQTT Client to a new thread, open the `Configuration.xml` with the RA Configuration. While ensuring the correct thread is selected on the left, use the tab for **Stacks > New Stack > Search** and search for the keyword AWS MQTT Client.

² The root CA certificate provided by AWS IoT is signed by Digital Guardian.

³ Public Key length used is 2048 bits.

⁴ Recommended for local server testing and not for production/deployment.

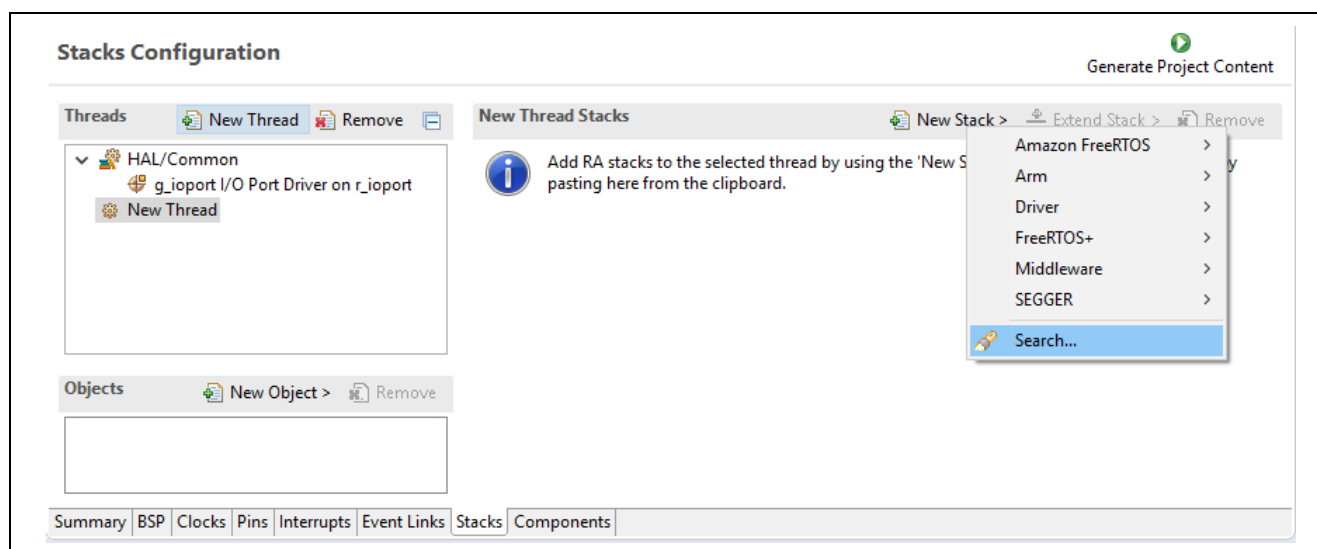


Figure 2. AWS MQTT Client Module Selection

Adding the AWS MQTT Client Stack results in the default configuration with *some unmet dependencies*, as shown in the following graphic.

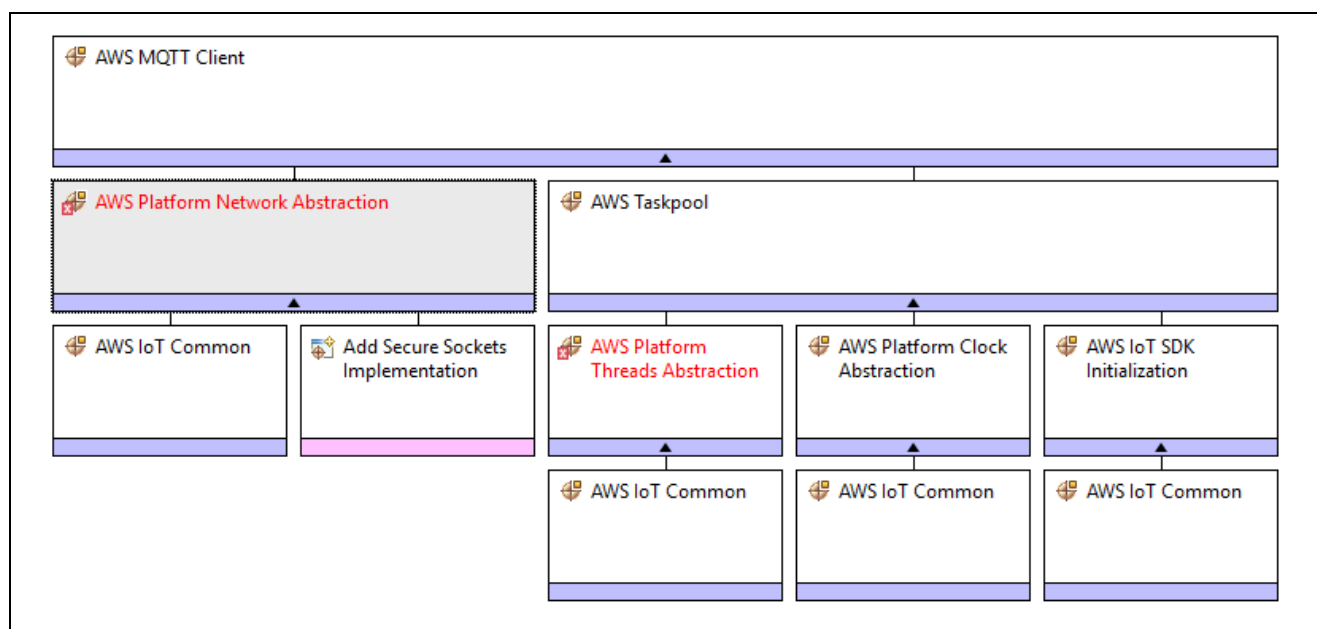


Figure 3. AWS MQTT Client Stack View

While the AWS MQTT Client stack shown contains a lot of dependencies and configurable properties, most default settings can be used as-is. The following changes are needed to meet all unmet dependencies (marked in red) for the AWS MQTT Client stack added to a new project (as shown above):

1. Enable Mutex and Recursive Mutex usage support as needed by IOT SDK and FreeRTOS in the created Thread properties.
2. Optionally, adjust the AWS IoT Common properties for 'IoT Thread Stack Size' and 'IoT Network Receive Task Stack Size'.

Upon completion of the two steps above, the AWS MQTT Client is ready to accept a Secure Socket Implementation, which has dependencies on using a TLS Session and an underlying TCP/IP implementation.

Additional documentation on the AWS MQTT Client is available in the FSP User's Manual under RA Flexible Software Package Documentation > API Reference > Modules > AWS MQTT

3. Secure Sockets Implementation

The AWS Secure Sockets module provides an API that is based on the widely used BSD Sockets. While the RA FSP contains a Secure Socket Implementation for both WiFi and Ethernet, this application primarily focuses on the use of an external Silex SX-ULPGN PMOD WiFi module.

Secure Sockets can be added to the Thread Stack by clicking on **Add Secure Sockets Implementation > New > Secure Sockets on WiFi**.

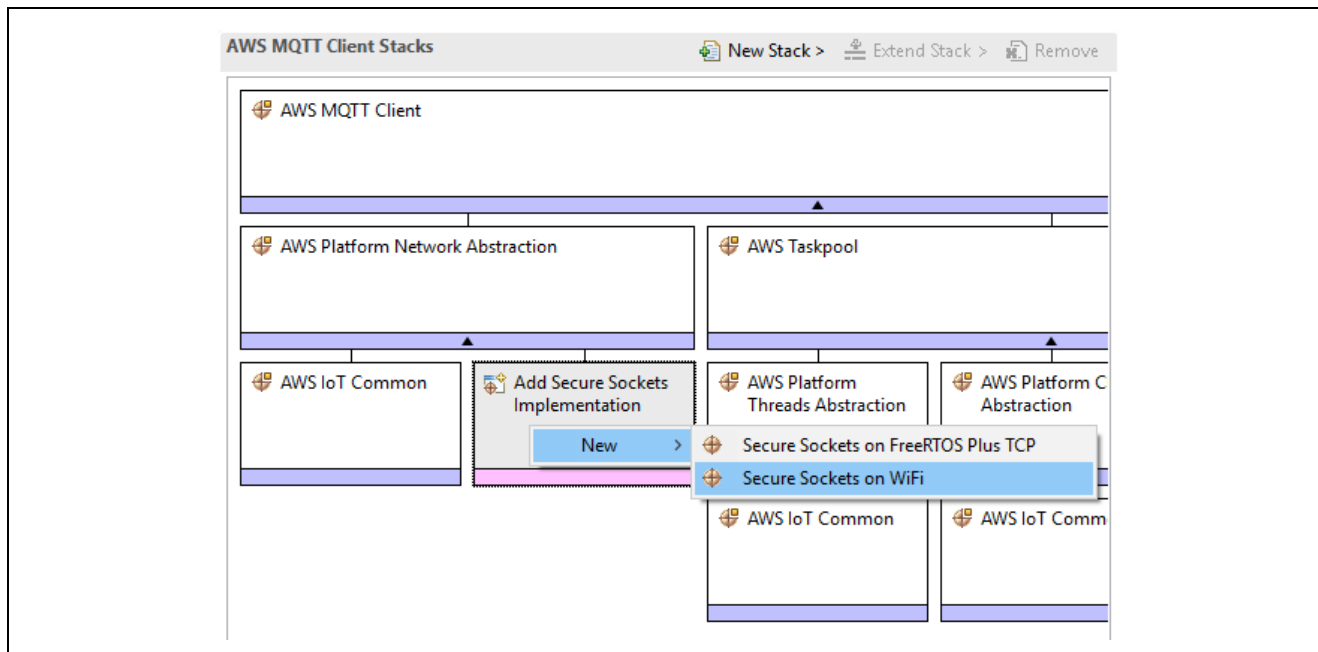


Figure 4. Adding Secure Socket to the MQTT Client Module

Upon addition, the needed stack is complete and has unmet dependencies for the dependent modules. Here in this case the flash file system for the persistent storage is needed. This can be added by clicking on **Add AWS PKCS11 PAL > New > AWS PKCS11 PAL on LittleFS**.

The added stack has unmet configuration for the sub-components, highlighted in red in the next figure that should be addressed.

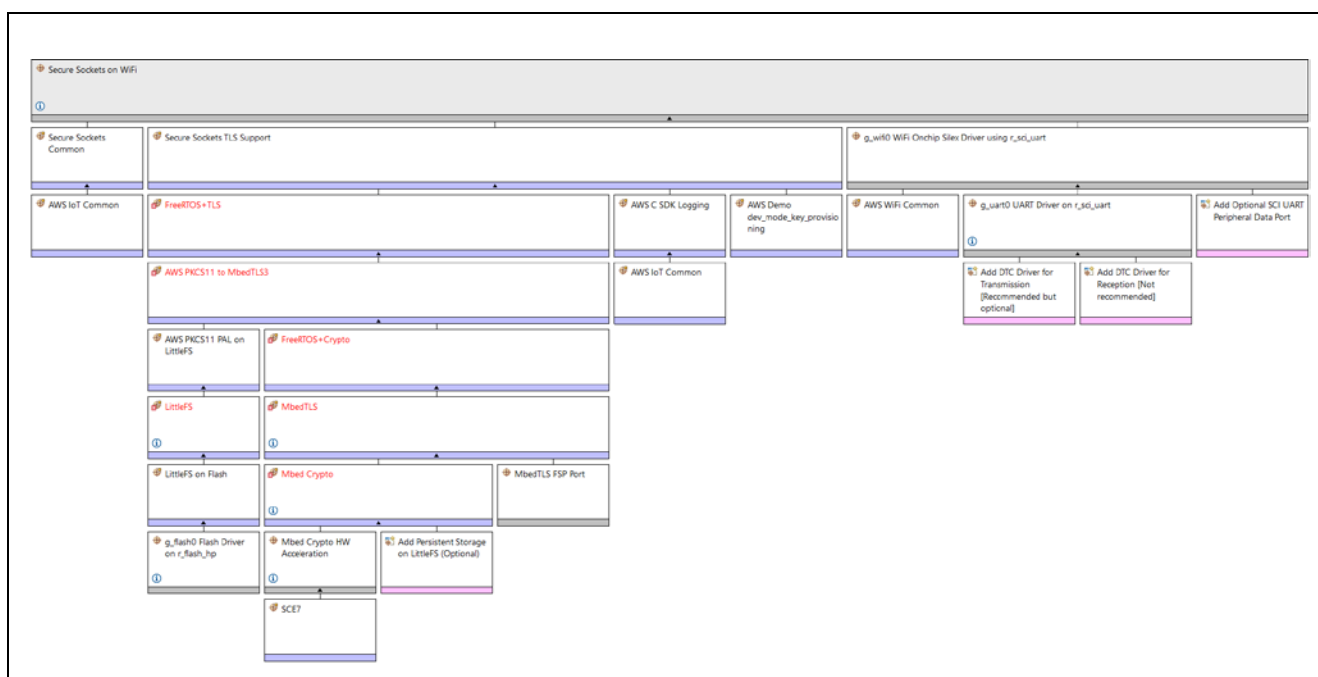


Figure 5. Expanded Secure Socket Module

Satisfying most unmet dependencies as identified by the RA Configurator is recommended, however, some are optional. For example, this application project uses one socket/UART for communication with the Silex WiFi Pmod™. As a result, the property for **Number of supported sockets instances** should be set to 1. This property is found by under Common Properties for WiFi on-chip Silex Driver.

Note: To support multiple socket instances on this WiFi module, a second UART on the Module needs to be enabled. For the purpose of this Application Note, one socket is enough.

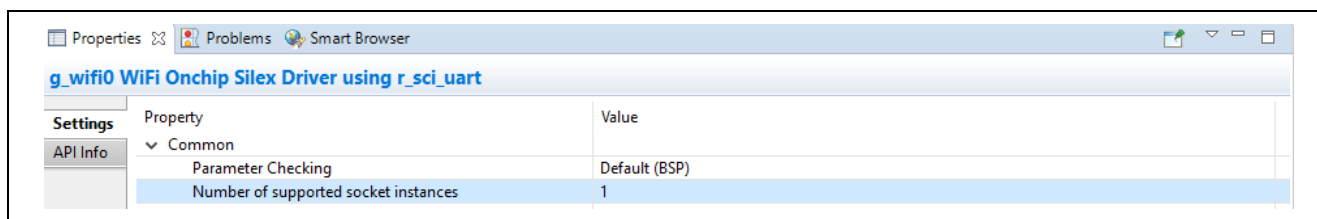


Figure 6. Socket Instances Selection

Remaining unmet dependencies which need to be addressed are related to the TLS stack discussed next.

Additional documentation on the AWS MQTT Client is available in the FSP User's Manual under RA Flexible Software Package Documentation > API Reference > Modules > AWS Secure Sockets.

4. Mbed TLS

mbedtls is ARM's implementation of the TLS protocols as well as the cryptographic primitives required by those implementations. mbedtls is also solely used for its cryptographic features even if the TLS/SSL portions are not used.

Secure Socket TLS Support uses FreeRTOS+TLS, which eventually uses mbedtls. Use of mbedtls requires configuration and operation of Mbed Crypto module which in turn operates the SCE on the MCU.

The following underlying mandatory changes are needed to a project using the Secure Sockets on FreeRTOS+TLS module:

1. Use FreeRTOS heap implementation scheme 4 (first fit algorithm with coalescence algorithm) or scheme 5 (first fit algorithm with coalescence algorithm with heap spanning over multiple non adjacent/non-contiguous memory regions) [11].
2. Enable support for dynamic memory allocation in FreeRTOS.
3. Enable mbedtls platform memory allocation layer.
4. Enable the mbedtls generic threading Layer that handles default locks and mutexes for the user and abstracts the threading layer to use an alternate thread-library.
5. Enable Elliptic Curve Diffie Helleman library.
6. Change FreeRTOS Total Heap Size to a value greater than 0x1500.

Additional documentation on the AWS MQTT Client is available in the FSP User's Manual under RA Flexible Software Package Documentation > API Reference > Modules > Crypto Middleware (rm_psa_crypto).

5. MQTT Module APIs Usage

The AWS MQTT Client is documented online [12]. The following table lists APIs provided by AWS MQTT Client that are used as a part of the Application Example.

Table 2. MQTT Module APIs

lotMqtt_Connect	Establish a new MQTT connection.
lotMqtt_Init	One-time initialization function for the MQTT library.
lotMqtt_IsSubscribed	Check if an MQTT connection has a subscription for a topic filter.
lotMqtt_OperationType	Returns a string that describes an lotMqttOperationType_t.
lotMqtt_Publish	Publishes a message to the given topic name and receive an asynchronous notification when the publish completes
lotMqtt_ReceiveCallback	Network receive callback for the MQTT library.
lotMqtt_strerror	Returns a string that describes an lotMqttError_t.
lotMqtt_Subscribe	Subscribes to the given array of topic filters and receive an asynchronous notification when the subscribe completes.
lotMqtt_TimedSubscribe	Subscribes to the given array of topic filters with a timeout.
lotMqtt_TimedUnsubscribe	Unsubscribes from a given array of topic filters with a timeout.
lotMqtt_Unsubscribe	Unsubscribes from the given array of topic filters and receive an asynchronous notification when the unsubscribe completes.
lotMqtt_Wait	Waits for an operation to complete.
lotMqtt_Disconnect	Closes an MQTT connection

6. Cloud Connectivity Application Example

6.1 Overview

This Application example demonstrates the usage of API available through the Renesas FSP-integrated modules for Amazon IoT SDK C, Mbed TLS module, Amazon FreeRTOS and HAL Drivers operating on Renesas RA MCUs. Network connectivity is established using the Silex PMOD Wi-Fi Module. The application running on a Renesas Evaluation Kit also serves as a reference system for the operation of MQTT Client, Mbed TLS/Crypto, Wi-Fi Module configuration using the FSP configurator. The application may be used as a starting point for inspiring other customized Cloud-based solutions using Renesas RA MCUs. In addition, it marginally demonstrates the operation and setup of cloud services available through the cloud service provider.

The upcoming sub-sections show step-by-step creation of a device and security credentials policies as required by the AWS IOT on the cloud side to communicate with the end devices. The example accompanying this documentation, demonstrates Subscribe and Publish messaging between and MQTT Client and MQTT Broker, periodic publication of Temperature data, asynchronous publication of "User Push Button" event from the MCU to the Cloud. The device is also subscribed to receive actuation events (LED ON/OFF) from the Cloud, thereby showing two-way control.

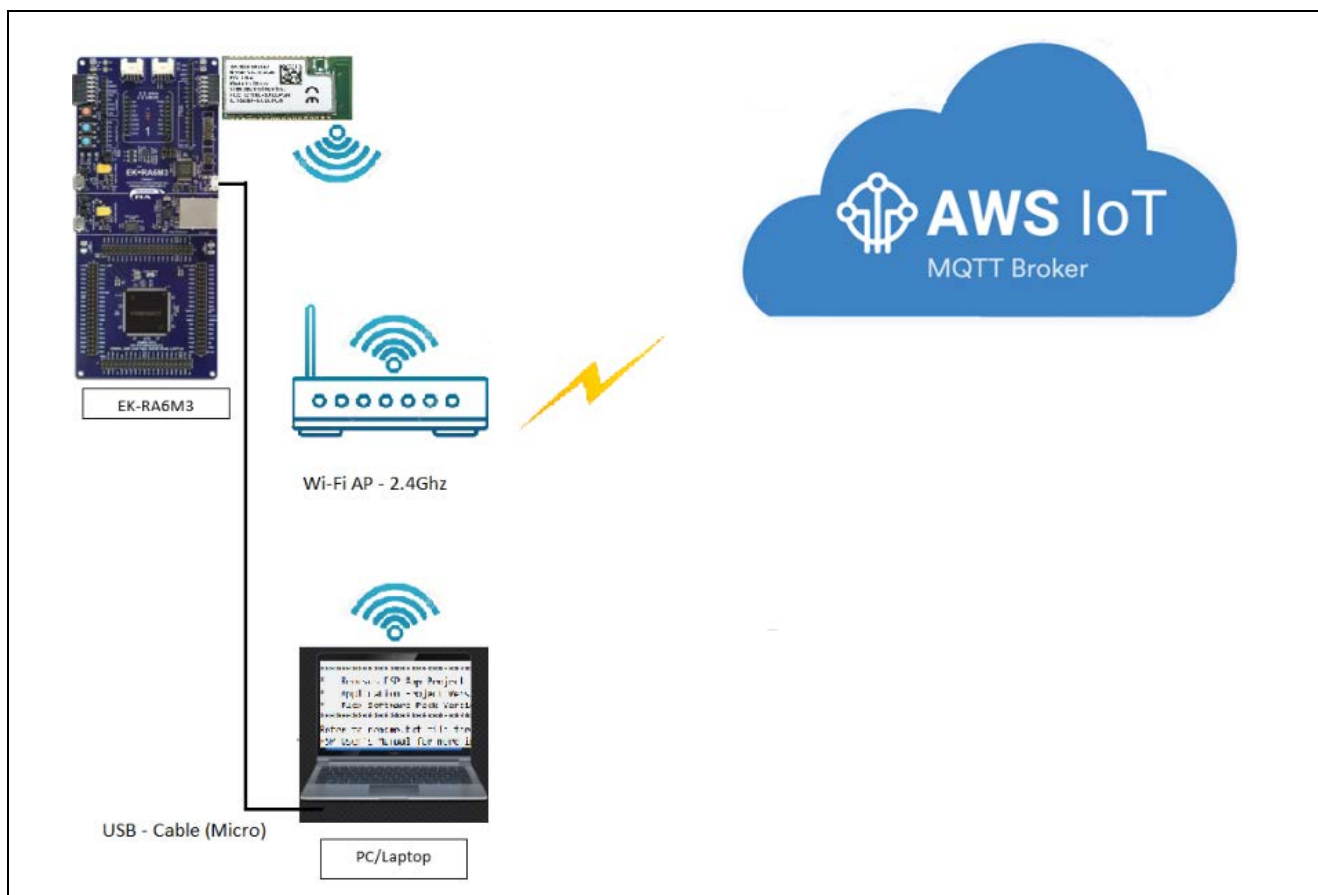


Figure 7. Application Project High level Overview

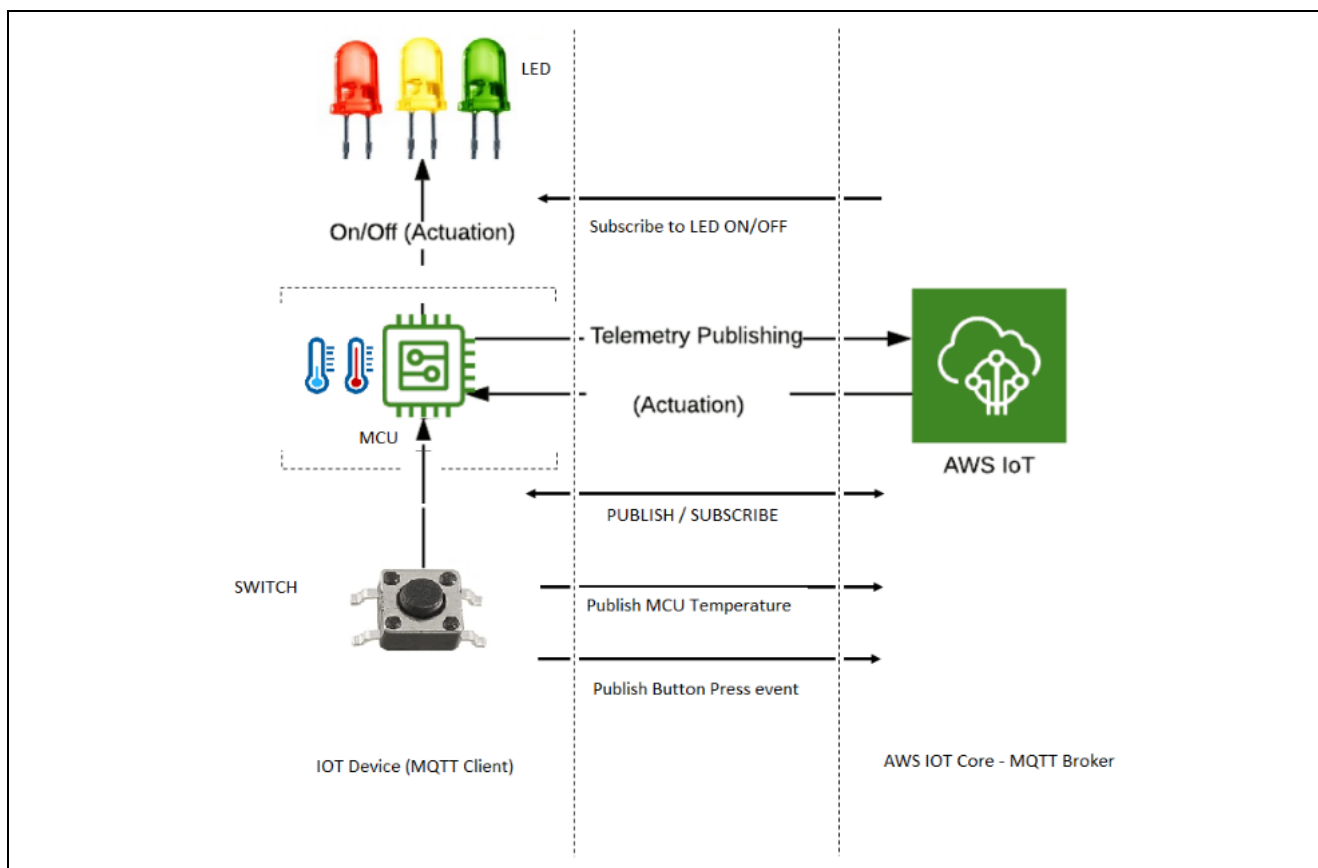


Figure 8. MQTT Publish/Subscribe to/from AWS IOT Core

6.2 MQTT/TLS Application SW Architecture Overview

The following files from this application project serve as a reference:

Table 3. Files Used in Application Project

No.	Filename	Purpose
1	src/application_thread_entry.c	Contains data structures functions and main thread used in Cloud Connectivity application.
2	src/aws_clientcredential.h ⁵	Required for compilation only. File to be populated with information unique for each client connecting to AWS. See [13]. User is requested to adjust usr_config.h instead.
3	src/aws_clientcredential_keys.h ⁵	Required for compilation only. File to be populated with information unique for each client connecting to AWS. See [13]. User is requested to adjust usr_config.h instead.
4	src/common_utils.h	Contains macros, data structures, and functions commonly used across the project.
5	src/hal_entry.c	Unused file automatically generated by FSP. This file is used for non-RTOS based projects.
6	src/itm_write.c	Re-routes logging information to Cortex-M Instrumentation Trace Macrocell
7	src/mqtt_interface.c	Contains data structures and functions used in mqtt interface for Cloud Connectivity
8	src/mqtt_interface.h	Accompanying header for exposing functionality provided by mqtt_interface.c
9	src/SEGGER_RTT/SEGGER_RTT.c	Implementation of SEGGER real-time transfer (RTT) which allows real-time communication on targets which support debugger memory accesses while the CPU is running.
10	src/SEGGER_RTT/SEGGER_RTT.h	
11	src/SEGGER_RTT/SEGGER_RTT_Conf.h	
12	src/SEGGER_RTT/SEGGER_RTT_printf.c	
13	src/usr_config.h	To customize the user configuration to run the application.
14	src/usr_hal.c	Contains data structures and functions used for the Hardware Abstraction Layer initialization and associated utilities
15	src/usr_hal.h	Accompanying header for exposing functionality provided by usr_hal.c
16	src/usr_wifi.c	Contains data structures and functions used to operate the WiFi module.
17	src/usr_wifi.h	Accompanying header for exposing functionality provided by usr_wifi.h
18	src/usr_app.h	Accompanying header file for the application_thread.

⁵ Successful connection requires that unique values are populated into constants in this file.



Complete steps to create the Project from the start using the e² studio and FSP configurator. The table below shows the step-by-step process in creating the Project. It is assumed that the user is familiar with the e² studio and FSP configurator. Launch the installed e² studio for the FSP.

	Step	Intermediate Steps
1	Project Creation	File → New → RA C/C++ Project
2	Project Template	Templates for New RA C/C++ Project → Renesas RA C Executable Project → Next
3	e² studio - Project Configuration (RA C Executable Project) →	Project Name (Name for the Project)
	Toolchains	(Default: GNU ARM Embedded) → Next
4	Device Selection →	FSP Version: 1.1.0
		Board: EK-RA6M3
		Device: R7FA6M3AH3CFC
		RTOS: FreeRTOS
5	Select Tools	Toolchain: GNU ARM Embedded (Default)
		Toolchain version: (9.2.1.20191025)

		Debugger: J-Link ARM Next→
6	Project Template Selection	FreeRTOS - Minimal - Static Allocation → Finish
7	Stacks Tab (Part of the FSP Configurator)→	Threads → New Thread
8	Config Thread Properties→	
	Common → General	Max Task Name Len: 32
	Thread->	Symbol: Application thread
		Name: Application thread
		Stack size: 8192 Bytes
		Priority: 3
		Thread Context: NULL
		Memory Allocation: Static
9	Generic RTOS configs under thread (Additional configuration on top of the Default Config provided by FSP)	
	Common → General	Use Mutex: Enabled
		Use Recursive Mutexes: Enabled
	Common → Memory Allocation	Support Dynamic Allocation: Enabled
		Total Heap Size: 0x20000
10	Add the Heap Implementation in HAL/Common	
	New Stack →	FreeRTOS → Memory Management→ Heap 4
11	Adding the MQTT Client Module to the Thread	
	Note: Now the Newly created thread (Application thread) is ready to add new Stack (Here the MQTT Client is added)	
	New Stack →	FreeRTOS → Libraries → AWS MQTT Client
12	Configuring the “AWS IoT Common” Module (Additional configuration on top of the Default Config provided by FSP)	
	Properties → Common →	Platform Name: "AWS Cloud Connectivity".
13	Adding the Secure Socket Implementation. Implementation is available for 1) WiFi (Secure Socket on WiFi) and 2) Ethernet (Secure Socket on FreeRTOS Plus TCP).	
	Add Secure Sockets Implementation →	New → Secure Socket on WiFi
14	Adding Persistent storage support for AWS PKCS11 and resolve the error in the configurator by selecting the Heap size in the BSP Tab	
	Add AWS PKCS11 PAL module ->	New -> AWS PKCS11 PAL on LittleFS
	BSP->RA Common->	Heap size : 0x1000
15	Some dependency related to TLS Support are needed to be resolved to remove the error in the FSP configurator by modifying the “Mbed Crypto” Property Settings.	
	Common → Platform →	MBEDTLS_PLATFORM_MEMORY : Define
	Common → General →	MBEDTLS_THREADING_C : Define
	Common → General →	MBEDTLS_THREADING_ALT : Define
	Common → Public Key Cryptography (PKC)→	ECC → MBEDTLS_ECDH_C : Define

16	Resolve the dependency for the WiFi Onchip Silex Driver using “r_sci_uart” Module.	
	Note: For Multiple socket connection using this WiFi Module, 2 UARTS needs to be used. Even though the WiFi Module provides 2 UARTs (For Multiple Sockets) The PMOD connector only exposes 1 UART connectivity. Therefore, in this case, if the requirement is to use in the multi socket mode, proper wiring needs to be taken care to route the UART connection the WiFi Module. Note: In case of MQTT Connectivity example project, the connections can be managed with 1 Socket connection only, and hence the socket instance is chosen as 1.	
	Configuring the WiFi Onchip Silex Driver using r_sci_uart	
	Common →	Number of supported Socket instances: 1
		Module Reset Port: 08
Module Reset Pin: 00		
17	UART Driver r_sci_uart configuration UART driver needs to be configured based on the data transmission and receive requirements such as speed, FIFO, DTC support, Flow control, handling of UART interrupts etc.	
	Common →	FIFO Support: Enable
		DTC Support: Enable
		Flow control Support: Disable
	Module Driver →	General → Channel: 9 (For PMOD 1)
		Baud → Baud Rate: 115200
		Flow Control →
		CTS/RTS Selection: RTS(CTS is disabled)
		Flow Control → Pin control: Disabled
		Flow Control → RTS Port: Disabled
		Flow Control → RTS Pin: Disabled
		Interrupts→
		Receive Interrupt Priority: Priority 5
		Interrupts →
		Transmit Data Empty Interrupt Priority: Priority 5
		Interrupts →
		Transmit End Interrupt Priority: Priority 5
		Interrupts →
		Error Interrupt Priority: Priority 5
18	Add DTC driver to complete the DTC Driver support for the UART	
	Add DTC driver for Transmission →	New → Transfer Driver on r_dtc
	Add DTC driver for Reception →	New → Transfer Driver on r_dtc
19	Adding the HAL Modules as required for the Application Project: Here, ADC, Timer0, Timer1, ELC, and External IRQ Modules are used for MCU Temperature, 30 Seconds periodic timer, 1 second Periodic Heartbeat Monitor Timer, Event linking of ADC Data read and Push button switches respectively.	
	HAL/Common Stacks → New Stack	Driver→Input→External IRQ Driver on r_icu
	Property Settings for r_icu	Name: pushButtonS1
		Channel: 13
		Trigger: Rising
		Digital Filtering: enabled
		Digital Filtering Sample Clock (PCLK/64)
		Pin Interrupt Priority: Priority 10

		Callback: pb_callback
	HAL/Common Stacks → New Stack	Driver→Input→External IRQ Driver on r_icu
	Property Settings for r_icu	Name: pushButtonS2
		Channel: 12
		Trigger: Rising
		Digital Filtering: enabled
		Digital Filtering Sample Clock (PCLK/64)
		Pin Interrupt Priority: Priority 10
		Callback: pb_callback
	HAL/Common Stacks → New Stack	Driver→Timers→Timer Driver on r_gpt
	Property Settings for r_gpt→General	Name: gpt
		Channel: 0
		Mode: Periodic
		Period: 30
		Period Unit: Seconds
	Interrupts:	Callback: NULL
		Overflow/Crest Interrupt Priority: Priority 10
	HAL/Common Stacks → New Stack	Driver→Timers→Timer Driver on r_gpt
	Property Settings for r_gpt→General	Name: g_hb_timer
		Channel: 1
		Mode: Periodic
		Period: 1000
		Period Unit: MilliSeconds
	Interrupts	Callback: g_hb_timer_cb
		Overflow/Crest Interrupt Priority: Priority 10
	HAL/Common Stacks→New Stack	Drivers→System→ELC Driver on r_elc
	Property Settings for r_elc→Module g_elc Driver on r_elc	Name: g_elc
	HAL/Common Stacks→New Stack	Drivers→Analog→ADC Driver on r_adc
	Property Settings for r_adc→General	Name: adc
		Unit: 0
		Resolution: 12-bit
		Alignment: Right
		Clear after read: On
		Mode: Single Scan
		Double-trigger: Disabled
	Property Settings for r_adc→Input→	Channel Scan Mask : Temperature Sensor
	Property Settings for r_adc→Interrupt→	Normal/Group A Trigger: GPT0 COUNTER OVERFLOW(overflow)
		Callback: adc_mcu_temp_callback
		Scan End Interrupt Priority: Priority 10

20	Modifying the Pin configuration as required for the Application Projects (UART related Pins, Flow Control Pins, Disable Multi functionality pins as needed to use alternative pin functions). Note: The GLCDC Pins are shared with SCI9 (PMOD 1), In this Application Project GLCDC is not used, so disable it.	
	Pins Tab in the FSP configurator→	Peripherals→ Graphics: GLCDC→GLCDC0
		Operation Mode: Disabled
		Peripherals→Connectivity: SCI → SCI9
		Pin Group Selection: _A only
		Operation Mode: Custom
		TXD_MOSI : P203
		RXD_MISO: P202
		SCK : None
		CTS_RTS_SS: None
		SDA: None
		SCL: None
21	Modifying the BSP Settings - RA Common for (Main stack and Heap Settings)	
	Property Settings for RA Common	Main stack size(bytes) : 0x400
		Heap size (bytes) : 0x1000
22	Adding FreeRTOS Objects for the Application (Topic Queue needs to be created for the application – Message Queue)	
	Stacks Tab → Objects →	New Object → Queue
	Property Settings for the Queue	Symbol: g_topic_queue
		Item Size (Bytes): 72
		Queue Length (Items): 20
		Memory Allocation: Static

The above configuration is a prerequisite to generate the required Stack and features for the Cloud connectivity application provided with this app note. Once the **Generate Project Content** button is clicked, it generates the source code for the project. The generated source code is the required drivers, stack and middleware. The user application files are required to be added into the `src` folder.

Note: FSP generated Code needs to be called/used from the application, while some of the middleware needs to be called exclusively as part of the application for proper initialization. For instance “`mbed_tls_setup()`” call initializes the SCE and TRNG; `SYSTEM_Init()` initializes and prepares the Crypto and Socket Libraries and `IoTSdk_Init()` initializes and prepares the IOT libraries.

For the validation of the created project, the same source files listed in the section MQTT/TLS Application SW Architecture Overview (Table 3) may be added.

The Details of the configurator from the default settings to changed settings are described in the following sections with reason of change.

6.4 MQTT/TLS Configuration

Describes the MQTT and TLS module configuration settings that are done as part of this application example.

The following table lists changes made to a default configuration populated by the RA Configurator.

Table 5. Default Configuration

Property	Original Value	Changed Value	Reason for change
Application Thread			
Common > General > Use Mutexes	Disabled	Enabled	This requirement is set by the AWS IOT SDK C stack
Common > Memory Allocation > Support Dynamic Allocation	Disabled	Enabled	This requirement is set by the AWS IOT SDK C stack
Common > Memory Allocation > Total Heap Size	0	0x20000	Heap required for the FreeRTOS, AWS IOT SDK, Mbed TLS
AWS IoT Common			
Common > IoT Thread Default Stack Size	512	4096	This stack size is for the internal IoT thread. The Application requires more stack than the default size.
Common > IoT Network Receive Task Stack Size	512	4096	This stack size is for the internal Network Receive thread. The Application requires more stack than the default size.
Platform Name	Unknown	AWS Cloud Connectivity	This value is user selectable and can be set to any value.
Wi-Fi On-chip Silex Driver			
Common > Number of Socket Instances Supported	16	1	For MQTT, only 1 socket is needed. However, to use more than 1 socket, a second UART must be opened and used. Currently in order to use the Silex Module with multiple sockets, additional wiring would be required.
Module Reset Port	06	08	Supported Port on PMOD 1 changed from the default BSP provided Port.
Module Reset Pin	03	00	Supported Pin on PMOD 1 changed from the default BSP provided Pin.
UART Driver			
Common > FIFO Support	Disabled	Enabled	Allows buffering of received data, to avoid packet loss.
Common > DTC Support	Disabled	Enabled	Using the Data Transfer Controller reduces the CPU overhead required to transmit and receive information.

Common > RS232/RS485 Flow Control	Disabled	Disabled	Flow control is required for higher baud rate/data rates. This is Good practice to prevent the incoming data from overwhelming the receiver.
General > Channel	0	9	The PMOD 1 for the Silex Wi-Fi is physically connected to Serial Communication Interface channel 9 on EK-RA6M3/G
Interrupts > Receive Interrupt Priority	Priority 12	Priority 5	Data Receive and Transmit are given higher priority.
Interrupts > Transmit Data Empty Interrupt Priority	Priority 12	Priority 5	
Interrupts > Transmit End Interrupt Priority	Priority 12	Priority 5	
Interrupt > Error Interrupt Priority	Priority 12	Priority 5	
Transfer Driver for TXI	Not Used	Used	Use of the Data Transfer Controller reduces the CPU overhead required to transmit and receive information.
Transfer Driver for RXI	Not Used	Used	Use of the Data Transfer Controller reduces the CPU overhead required to transmit and receive information.
Mbed Crypto			
Platform > mbedtls_platform_memory	Undefine	Define	This selection is required in order to support the mbed_tls.
General > mbedtls_deprecated_removed	Define	Undefine	
General > mbedtls_check_params	Define	Undefine	
General > mbedtls_threading_alt	Undefine	Define	This selection is required in order to support the mbed_tls to plug in any thread library.
General > mbedtls_threading_c	Undefine	Define	This selection is required in order to support the mbed_tls to abstracts the threading layer to allow easy plugging in any thread-library.
Public Key Cryptography > ECC > mbedtls_ecdh_c	Undefine	Define	This selection is required in order to support the mbed_tls to enable the ECDH module.
LittleFS (Heap Selection)			
BSP ->RA Common Heap Size	0x0	0x1000	Heap selection for Heap 3 and below needs to be done here.

6.4.1 IoT Cloud Configuration (AWS)

6.4.1.1 AWS IoT Policies

AWS IoT Core policies are JSON (JavaScript Object Notation) documents that authorize a device to perform AWS IoT Core operations. AWS IoT defines a set of policy actions describing the operations and resources for which access can be granted or denied. For example:

- IoT: Connect represents permission to connect to the AWS IoT message broker.
- IoT: Subscribe represents permission to subscribe to an MQTT topic or topic filter.
- IoT: GetThingShadow represents permission to get a thing shadow.

JSON

JSON is an open standard, lightweight, data-interchange format. As a text document, it is easy for users to read and write, and for machines to parse and generate.

JSON is completely language independent, using conventions that are familiar to C-family programmers, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. The following example shows a JSON script used to turn on an LED.

```
{
  "state": {
    "desired": {
      "LED_value": "On"
    }
  }
}
```

AWS IoT Thing Shadow

A Thing Shadow (also referred to as a Device Shadow) is a JSON document used to store and retrieve current state information for a Thing (device, application, and so on).

The Thing Shadow service maintains a thing shadow for each thing connected to AWS IoT Core. Thing shadows may be used to get and set the state of a thing over MQTT or HTTP, regardless of whether the thing is connected to the Internet. Each thing shadow is uniquely identified by its name.

Amazon Web Services Signup

Amazon Web Services offers a free account (12 months) for each user. It is expected that a user account needs to be created on the AWS IoT Cloud service before continuing to the next section.

To create an AWS account, open to the following link in a web browser:

<https://portal.aws.amazon.com/billing/signup#/start>

Fill in the required details and create a user account.

Note: While creating the project, certificates and policies, the screenshots may look slightly different from what is shown in the document and users need to use navigation in the AWS IoT core environment to find corresponding attributes while working on this project.

6.4.2 Creating a Device on AWS IoT Core

The following steps detail how to create a device on the IoT Core user account. It is assumed that user account is created in the AWS IoT Core and the user has followed the AWS signup procedure.

6.4.2.1 Open AWS IoT Core Service

1. Connect to the AWS IoT service by typing **IoT Core** in the AWS services search bar.
2. Click **IoT Core**.

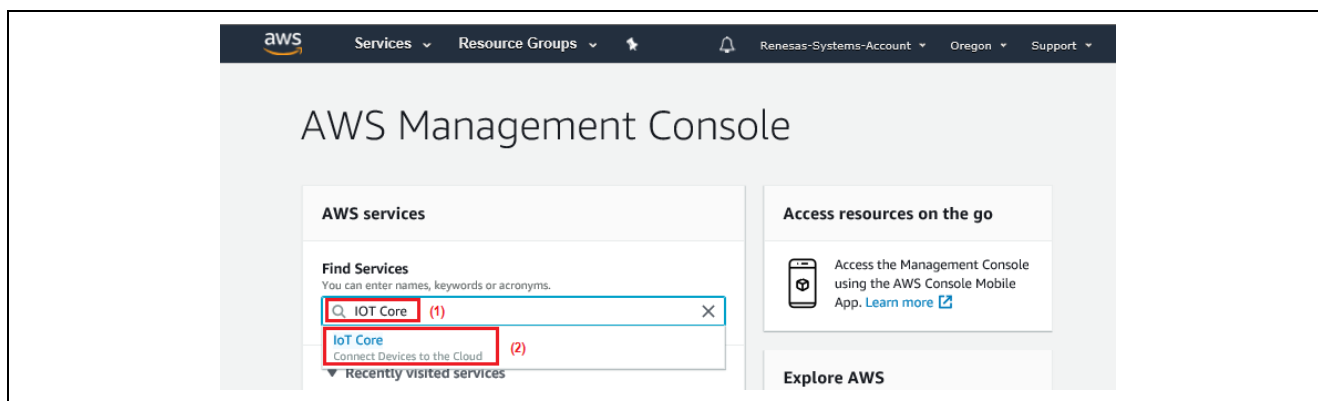


Figure 10. RA Cloud Connectivity AWS IoT Core Selection

6.4.2.2 Create a Thing

1. Start creating a device by selecting **Manage**. Be sure to select the appropriate region in the AWS console on the top right corner.

Note: A Thing created in one region will not be seen in another region.

2. Now select **Things**.
3. Next, select **Register a thing** to create a thing.

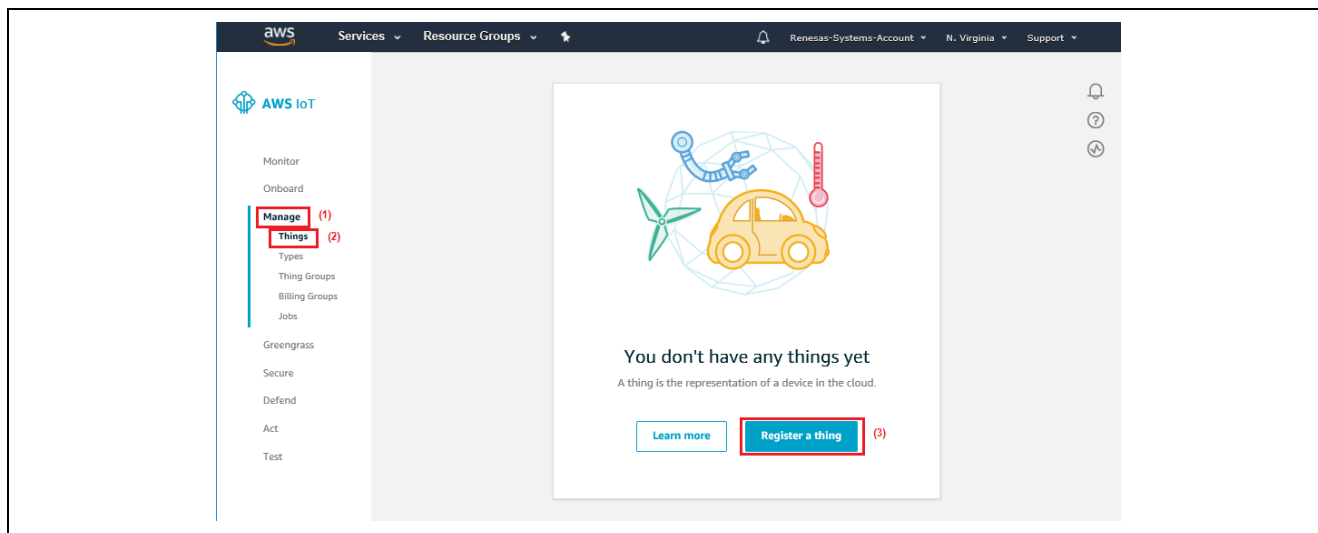


Figure 11. Register a Thing

4. Then select the **Create a single thing** button.

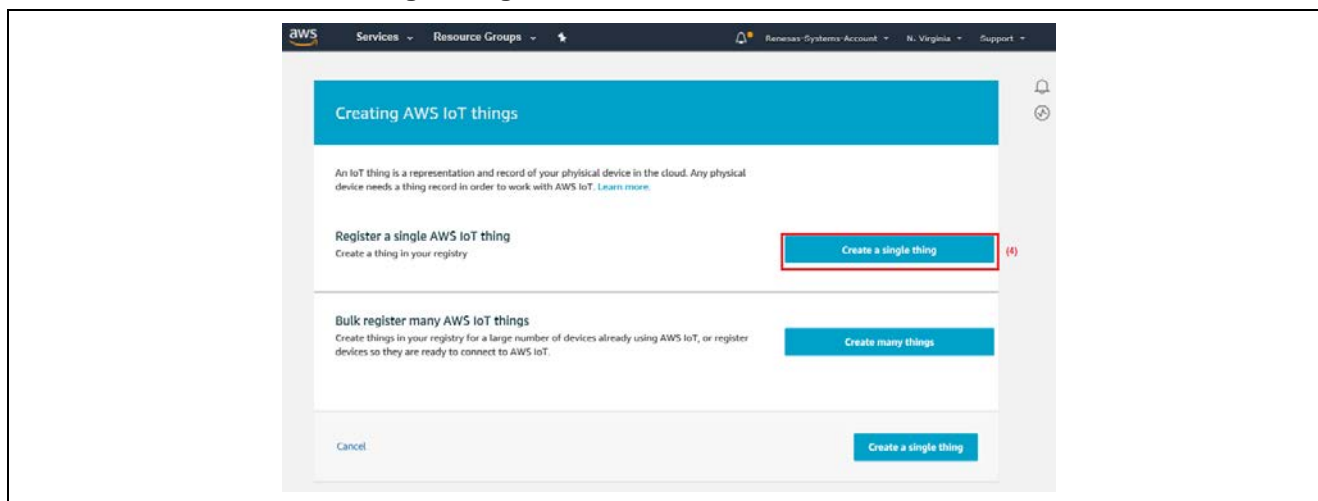


Figure 12. Create a Single Thing

5. Enter the **Thing Name**. In the example, a Thing by name **Thing_RA6** is being created.

Note: Remember to store the **Thing Name**. This information is required for future reference during configuration.

6. Create a Thing type by clicking the **Create a type** button. This will pop up another window

CREATE A THING STEP 1/3

Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

Thing_RA6 (5)

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

No type selected Create a type (6)

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Groups / Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key

Provide an attribute key, e.g. Manufacturer

Value

Provide an attribute value, e.g. Acme-Corporation Clear

Add another

Show thing shadow ▾

Cancel Back Next

Figure 13. Create a Type

7. Enter the **Type Name** and **Description**. Add the attributes by clicking the button **Add another** in the **Set searchable thing attributes** section.

Create a thing type

This will help you organize, categorize, and search for your things.

Name

Thing_RA6 7.1

Description

RA AWS Cloud Connectivity 7.2

Set searchable thing attributes

You can define up to three attributes for a thing type. Things associated with this type can be searched by using these fields.

Add another 7.3

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name	Value	
Provide a tag name, e.g. Manufacturer	Provide a tag value, e.g. Acme-Corporation	Clear
Add another		

Cancel Create thing type

Figure 14. Add Attributes

8. Add the attribute key and click the **Create thing type** button.

Create a thing type

This will help you organize, categorize, and search for your things.

Name

Description

Set searchable thing attributes

You can define up to three attributes for a thing type. Things associated with this type can be searched by using these fields.

Attribute key

8.1 Remove

Add another

Tags

Apply tags to your resources to help organize and identify them. A tag consists of a case-sensitive key-value pair. [Learn more](#) about tagging your AWS resources.

Tag name	Value	
<input type="text" value="Provide a tag name, e.g. Manufacturer"/>	<input type="text" value="Provide a tag value, e.g. Acme-Corporation"/>	Clear
Add another		

Cancel **8.2** Create thing type

Figure 15. Create Thing Type

9. Select the **Thing Type** and enter the attribute value. Click the **Next** button.

CREATE A THING

STEP 1/3

Add your device to the thing registry

This step creates an entry in the thing registry and a thing shadow for your device.

Name

Thing_RA6

Apply a type to this thing

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

Thing_RA6 9.1

Create a type

Add this thing to a group

Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Groups /

Create group Change

Set searchable thing attributes (optional)

Enter a value for one or more of these attributes so that you can search for your things in the registry.

Attribute key

Temperature

Value

25 9.2

Set non-searchable thing attributes (optional)

You can use thing attributes to describe the identity and capabilities of your device.

Attribute key

Provide an attribute key, e.g. Manufacturer

Value

Provide an attribute value, e.g. Acme-Corporation

Clear

Add another

Show thing shadow ▾

Cancel

BackNext

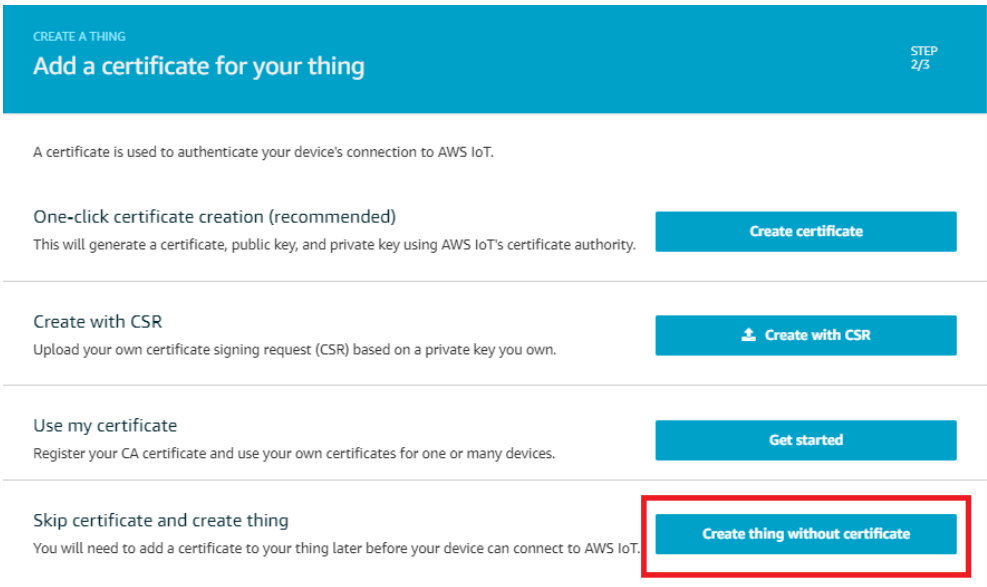
Figure 16. Create Attribute

R11AN0453EU0101 Rev.1.01
Jun.17.20

RENESAS

Page 25 of 38

10. Click the option **Create thing without certificate** to create a thing in AWS IoT.



CREATE A THING

STEP 2/3

Add a certificate for your thing

A certificate is used to authenticate your device's connection to AWS IoT.

One-click certificate creation (recommended)
This will generate a certificate, public key, and private key using AWS IoT's certificate authority.

Create certificate

Create with CSR
Upload your own certificate signing request (CSR) based on a private key you own.

Create with CSR

Use my certificate
Register your CA certificate and use your own certificates for one or many devices.

Get started

Skip certificate and create thing
You will need to add a certificate to your thing later before your device can connect to AWS IoT.

Create thing without certificate

Figure 17. Create Thing Without Certificate

6.4.3 Generating Device Certificate and Keys

At this point, it is assumed that the AWS IoT Thing has been created following the above instructions. Now, generate device certificates and keys for the AWS IoT Thing (Thing) created.

The Thing created appears in the **Things** section, as shown in the following screen.

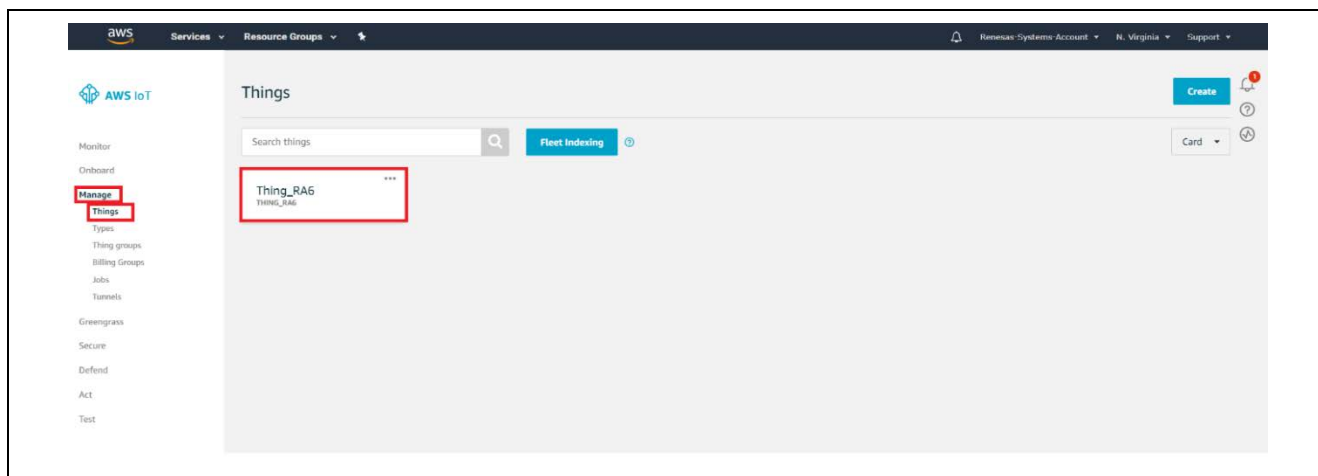


Figure 18. Created Thing

1. Click the Thing created. It will open in a new window with the Thing information.
In the example, the Thing created is called `Thing_RA6`.
2. Go to the **Security** tab and click **Create certificate** button, as shown in the following screen.

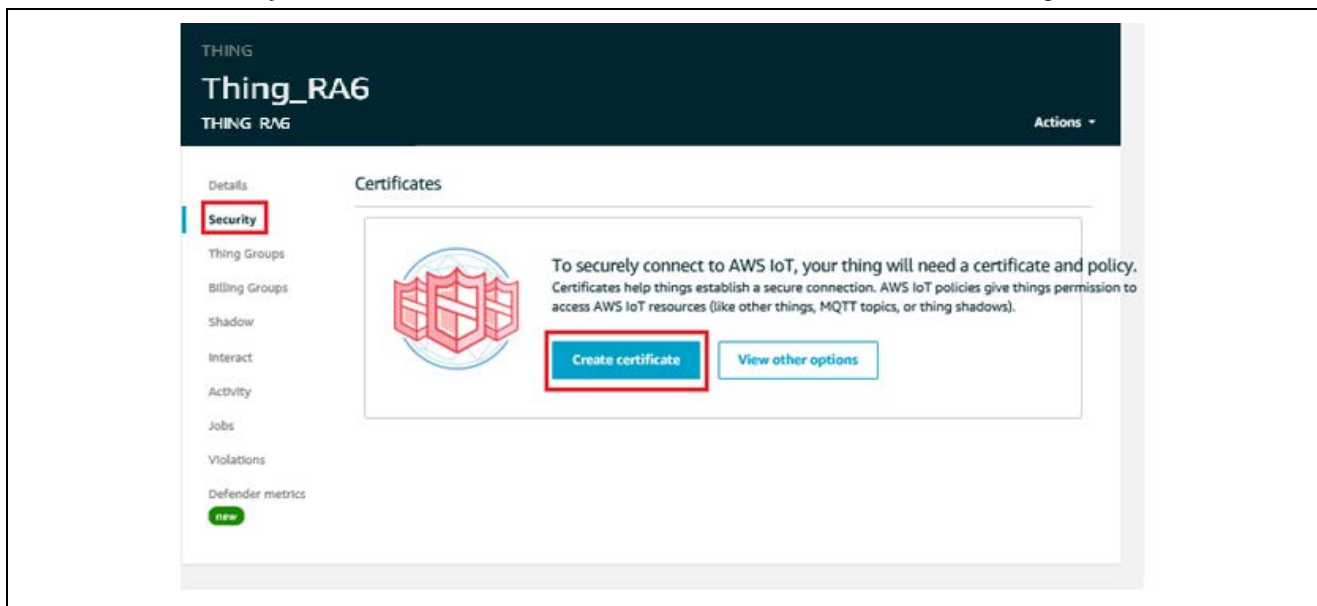


Figure 19. Create Certificate

It generates the following for the Thing created, as shown in the following screen.

- A device certificate,
- A public key,
- A private key,
- A root CA for AWS IoT

3. To download certificates, click the **Download** button next to each of the certificates and keys, as shown in the following screen.

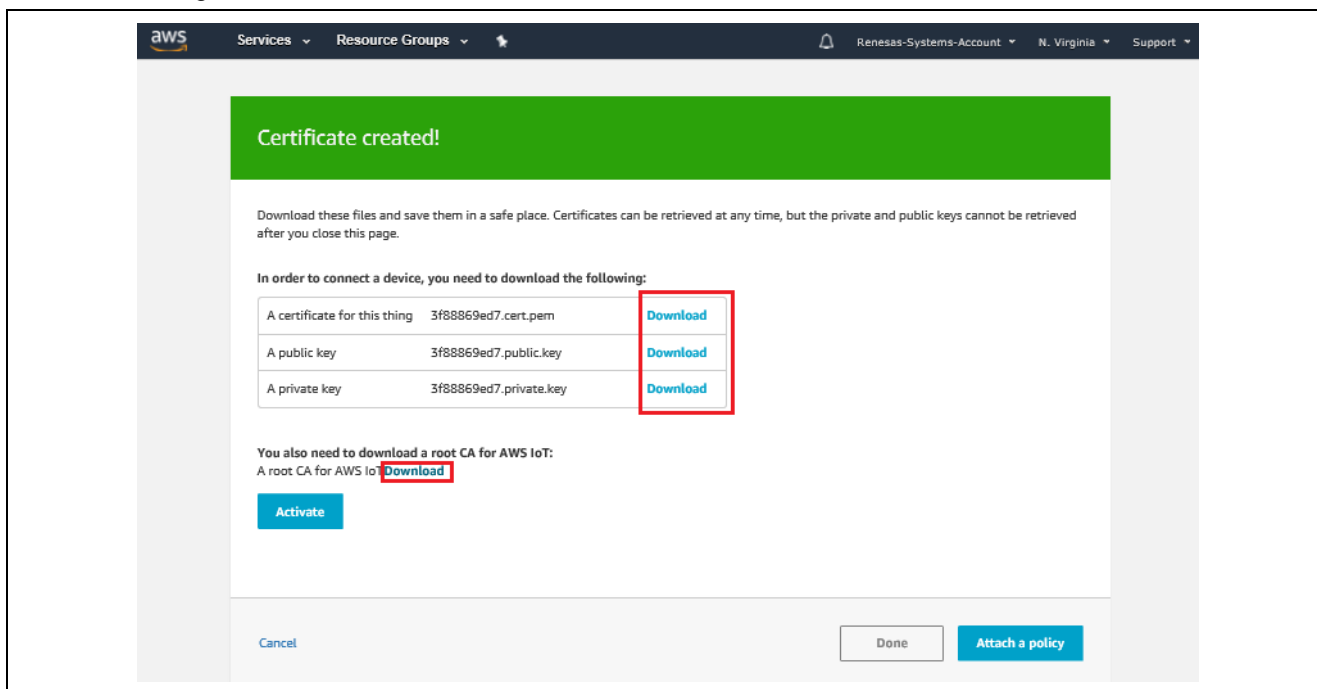


Figure 20. Download Certificate

4. Click the **Download** button. The root CA for AWS opens the link <https://docs.aws.amazon.com/iot/latest/developerguide/managing-device-certs.html#server-authentication>. Click **RSA 2048 bit key: Amazon Root CA 1**, as shown in the following figure to open the certificate in a new tab.

Note: Certificate generated by right clicking and downloading it to the PC might not work.

AWS Documentation » AWS IoT » Developer Guide » Security and Identity for AWS IoT » AWS IoT Authentication » X.509 Certificates » X.509 Certificates and AWS IoT

X.509 Certificates and AWS IoT

Client Authentication

AWS IoT can use AWS IoT-generated certificates or certificates signed by a CA certificate for device authentication. Certificates generated certificates signed by a CA certificate are set when the certificate is created.

Note

We recommend that each device be given a unique certificate to enable fine-grained management including certificate revocation. Devices must support rotation and replacement of certificates in order to ensure smooth operation as certificates expire.

To use a certificate that is not created by AWS IoT, you must register a CA certificate. All device certificates must be signed by the CA certificate.

You can use the AWS IoT console or CLI to perform the following operations:

- Create and register an AWS IoT certificate.
- Register a CA certificate.
- Register a device certificate.
- Activate or deactivate a device certificate.
- Revoke a device certificate.
- Transfer a device certificate to another AWS account.
- List all CA certificates registered to your AWS account.
- List all device certificates registered to your AWS account.

For more information about the CLI commands to use to perform these operations, see [AWS IoT CLI Reference](#).

For more information about using the AWS IoT console to create certificates, see [Create and Activate a Device Certificate](#).

Server Authentication

Server certificates allow your devices to verify that they're communicating with AWS IoT and not another server impersonating AWS IoT. : SDKs. AWS IoT server certificates are signed by one of the following CA certificates:

VeriSign Endpoints (legacy)

- RSA 2048 bit key: [VeriSign Class 3 Public Primary G5 root CA certificate](#)

Amazon Trust Services Endpoints (preferred)

- RSA 2048 bit key: **Amazon Root CA 1**
- RSA 4096 bit key: [Amazon Root CA 2](#)
- ECC 256 bit key: [Amazon Root CA 3](#)
- ECC 384 bit key: [Amazon Root CA 4](#)

Figure 21. Certificate Page

5. Go back to the AWS console and click the **Activate** button to activate the certificate just created.

6. After certificate activation, click the **Done** button to complete the certification/keys creation.

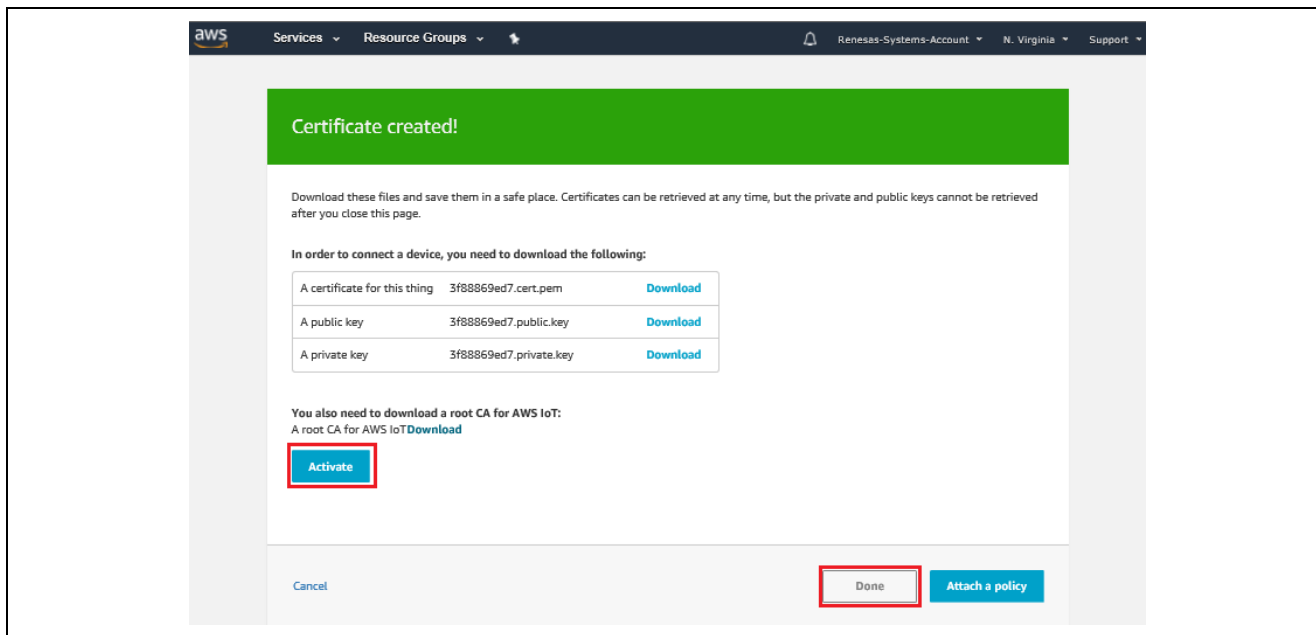


Figure 22. Certificate Created

Note: Since October 2018, AWS has recommended that all users create an Amazon Trust Services (ATS) endpoint and load these CA certificates onto their devices. The Amazon Root CA1 can be downloaded from: <https://docs.aws.amazon.com/iot/latest/developerguide/managing-device-certs.html>.

For users who created endpoints prior to October 2018 and are still using them to test this AP, it is recommended to use the `rootCA.pem` file given as part of this package.

6.4.4 Creating a Policy for a Device

To create a policy, go back to the Thing Hub.

1. Click the **Secure** option shown in the following screen.
2. Click the **Policies** option, which opens a window to create a new policy.
3. Click the **Create** button in the top right corner of the policies window to create new policy.

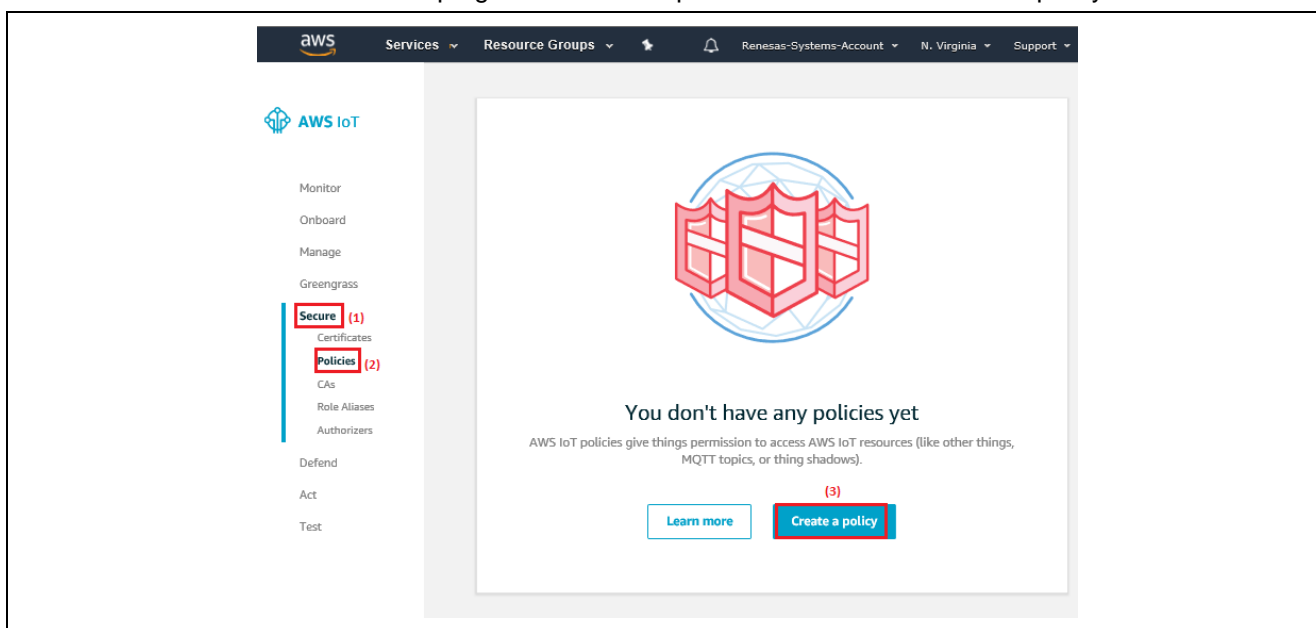


Figure 23. New Policy Creation

4. Enter the name for the policy in the **Name** box as shown in the following screen.
5. Under **Action**, type: **iot:***
6. Under **Resource ARN**, type: *****
7. Click **Allow**.
8. Click **Create**. The policy has now been created.

Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the [AWS IoT Policies documentation page](#).

Name
Thing_RA6_Policy 1

Add statements
Policy statements define the types of actions that can be performed by a resource. Advanced mode

Action
iot:* 2

Resource ARN
* 3

Effect
☒ Allow ☐ Deny 4

Add statement

Create 5

Figure 24. Create New Policy

6.4.5 Connecting the Certificate to the Policy

1. Click the **Secure** option as shown in the following screen. Then, click the **Certificates** option. It will open a window listing the device certificates created in AWS IoT Core service.
2. Choose the certificate created earlier for the Thing. This can be done by clicking the “...” in the top right corner of the certificate.
3. Click the **Attach policy** option from the drop-down menu.

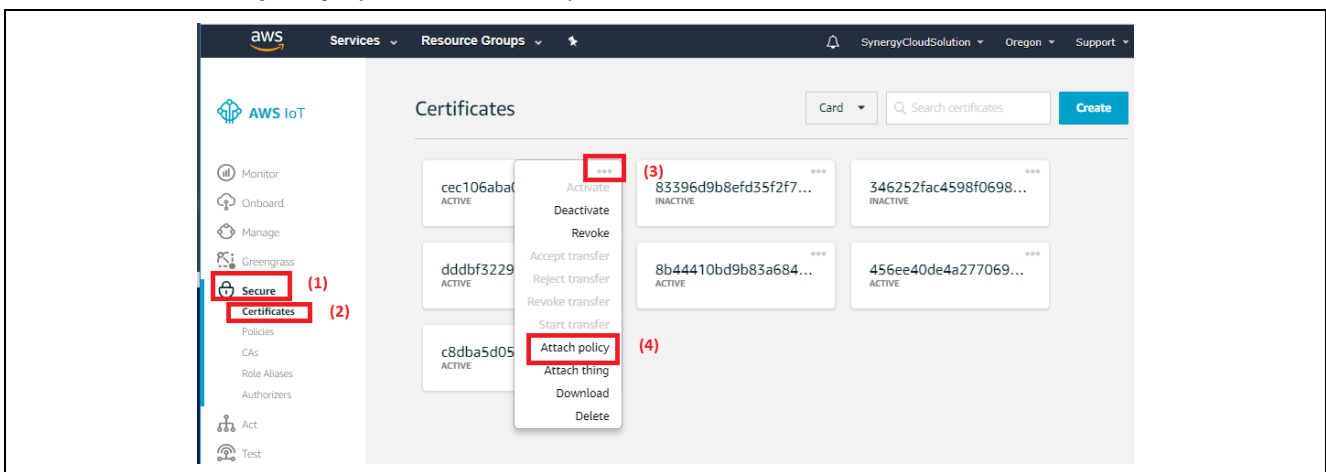


Figure 25. Connecting the Certificate to New Policy

4. Search for the policy on the **Search policies** window.
5. Choose the policy from the list and click the **Attach** button, as shown in the following screen.
6. The policy is now attached to the device certificate.



Figure 26. Attach Policies to Certificate

6.5 Running the MQTT/TLS Application Example

Note: If the project has been created with the instruction given in the previous sections using the FSP configurator, skip the importing of the Project and directly go to Loading the Executable Binary into the Target MCU. However, to quickly import and evaluate the application project archived with this document, browse the sub-sections below on Importing, building and loading sections.

6.5.1 Importing, Building and Loading the Project

6.5.1.1 Importing

This project can be imported into e² studio using instructions provided in the RA FSP User's Manual. See Section *Starting Development > e2 studio ISDE User Guide > Importing an Existing Project into e2 studio ISDE*.

6.5.1.2 Building the Latest Executable Binary

Upon successfully importing and/or modifying the project into e² studio IDE, follow instructions provided in the RA FSP User's Manual to build an executable binary/hex/mot/elf file. See Section *Starting Development > e2 studio ISDE User Guide > Tutorial: Your First RA MCU Project > Build the Blinky Project*.

Note: The attached Application Project Example may produce an error if test credentials are used during the run. Refer to the section Connecting to AWS IoT to enter the credentials for the device created per the section Creating a Device on AWS IoT Core.

6.5.2 Loading the Executable Binary into the Target MCU

The executable file may be programmed into the target MCU through any one of three means.

6.5.2.1 Using a Debugging Interface with e² studio

Instructions to program the executable binary are found in the latest RA FSP User Manual. See Section *Starting Development > e2 studio ISDE User Guide > Tutorial: Your First RA MCU Project > Debug the Blinky Project*.

This is the preferred method for programming as it allows for additional debugging functionality available through the on-chip debugger.

6.5.2.2 Using J-Link Tools

SEGGER J-Link Tools such as J-Flash, J-Flash Lite, and J-Link Commander can be used program the executable binary into the target MCU. Refer User Manuals UM08001, and UM08003 on www.segger.com.

6.5.2.3 Using Renesas Flash Programmer

The Renesas Flash Programmer provides usable and functional support for programming the on-chip flash memory of Renesas microcontrollers in each phase of development and mass production. The software supports all RA MCUs and the software user's manual is available online [15].

6.5.3 Powering up the Board

To connect power to the board, connect the USB cable to the EK-RA6M3 board's J10 connector (DEBUG_USB) and other end to the PC USB port, and run the debug application, using the following instructions.

Reset the board assembly associated with this application note to the default electrical jumper settings as specified in the board's documentation, that is, the hardware user's manual, before proceeding with the next set of instructions.

6.5.3.1 Deviation from Default Jumper Settings

The following are deviations from default board settings that should be performed prior to applying power to evaluate the application.

Table 6. Jumper Settings

Board Name	Jumper Settings
EK-RA6M3/G	No change necessary

6.5.3.2 Additional Components to Connect

This application project/note requires the user to connect the Silex SX-ULPGN-EVK WiFi evaluation kit [16] to a PMOD connector.

Table 7. Additional Components

Board Name	PMOD Identifier
EK-RA6M3/G	PMOD 1 (J26)

6.5.3.3 Power-on Behavior

Upon successful configuration and downloading of the image to the target RA MCU, the following behavior should be observed upon application of power.

1. The power LED on the RA MCU target assembly lights up.
2. The J-Link LED will be blinking based on the activity when it is connected.
3. The User LED (BLUE, GREEN, RED) are used to indicate the status of the application from the start of initialization to continuous status of running.

6.6 Connecting to AWS IoT

This section describes the steps to be followed to connect the device to the AWS IoT.

With the necessary infrastructure in place, it is necessary to input credentials for connecting to the 2.4 GHz WiFi Access Point and AWS IoT.

Note: Firewalls in the network may prevent connectivity to AWS IoT. Please configure the network to allow access to the MQTT Port 8883.

6.6.1 WiFi Credentials

Default Credentials for WiFi Connection are found in the file `usr_config.h`. These should be updated to match the credentials needed for the WiFi Access Point/WiFi Router that the board will connect to. The default credentials which need to be changed in the code are as shown below.

```
#define WIFI_SSID "Renesas"           "Change this to the correct network SSID"
#define WIFI_PW  "Renesas123"        "Change this to the correct network Password"
#define SECURITY "WPA2"               User Options are: "Open", "WEP", "WPA", "WPA2"
```

6.6.2 AWS IoT Credentials

Default Credentials for connectivity to AWS IoT are provided in the file `usr_config.h`. These should be updated to use the credentials generated per the guidelines provided in section Creating a Device on AWS IoT Core. MQTT ENDPOINT is obtained from the AWS IoT Core. The screenshots for the reference to capture the Endpoint info is given below.

```
#define USR_MQTT_ENDPOINT "aoh51vd4o23ku-ats.iot.us-east-1.amazonaws.com"
```

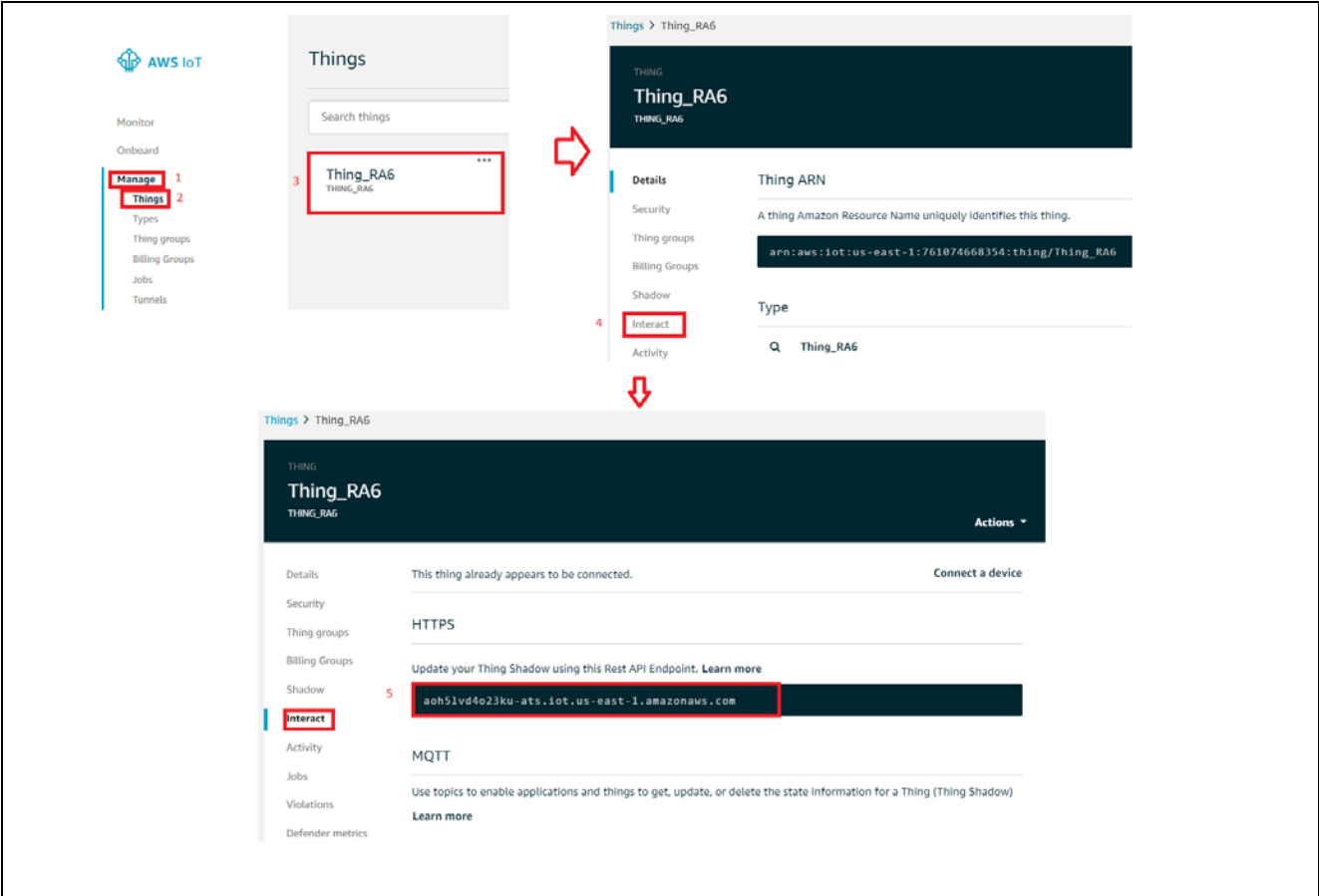


Figure 27. Getting User MQTT Endpoint

The downloaded asymmetric key pairs and certificates generated from AWS need to be included into the source code (`usr_config.h`) for the Application Example by converting each line into format required by the AWS SDK as shown below.

Downloaded Certificate/Key format from AWS	Desired Format for the AWS SDK
-----BEGIN PUBLIC KEY-----	"-----BEGIN PUBLIC KEY-----\n"

```
#define SERVER_CERTIFICATE "Populate the New Server Certificate"
#define CLIENT_CERTIFICATE "Populate the Client Certificate"
#define CLIENT_KEY "Populate the Client Key"
```

6.7 Verifying the Application Project

This section describes the steps on how to verify this application example functionalities.

Upon successfully programming the target RA MCU with the application example binary and powering up the board, a SEGGER J-Link RTT Console such as J-Link RTT Viewer should display output similar to output shown as follows:

Note: If the user wants to use the SWO viewer for the Console, change the default `#define USR_LOG_TERMINAL` (RTT_TERMINAL) to ITM_TERMINAL, and build the project.

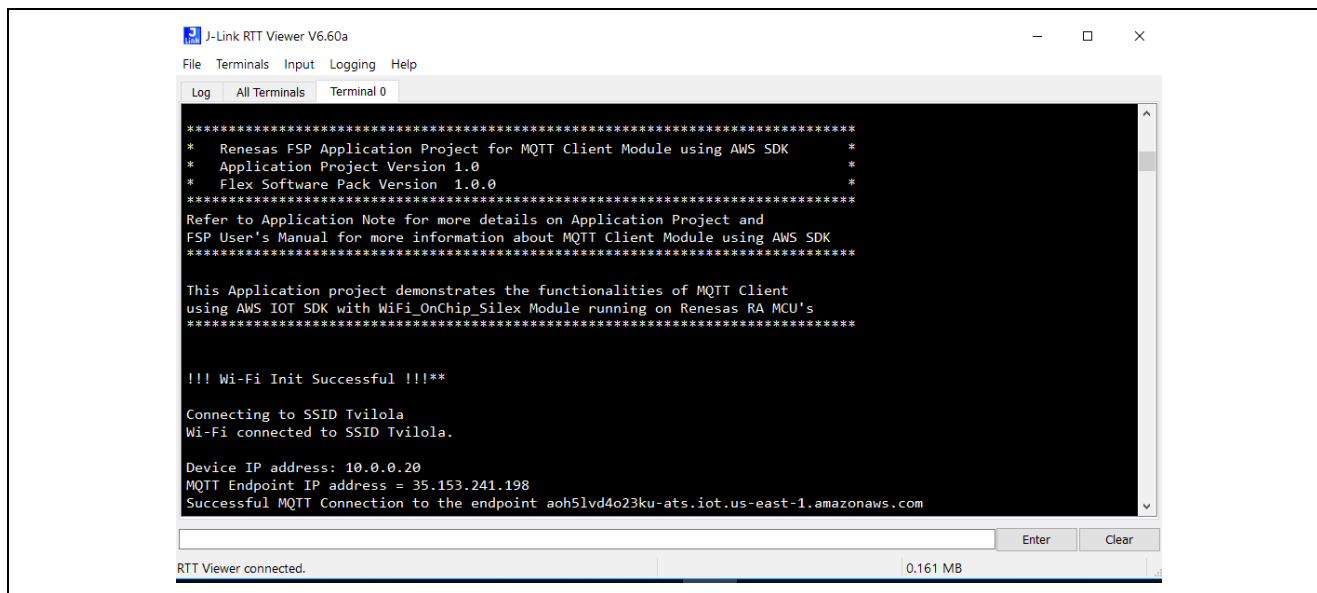


Figure 28. Welcome Screen on the Console

On the Cloud side, go to IoT Core and select the test tab. Subscribe to a topic (In this Application “aws/topic/temperature” and “aws/topic/switch_status” are the topics for subscription). You may observe events on the dashboard for the subscribed topics as shown below.

Note: Temperature messages are synchronous (every 30 seconds). Switch status are Asynchronous.

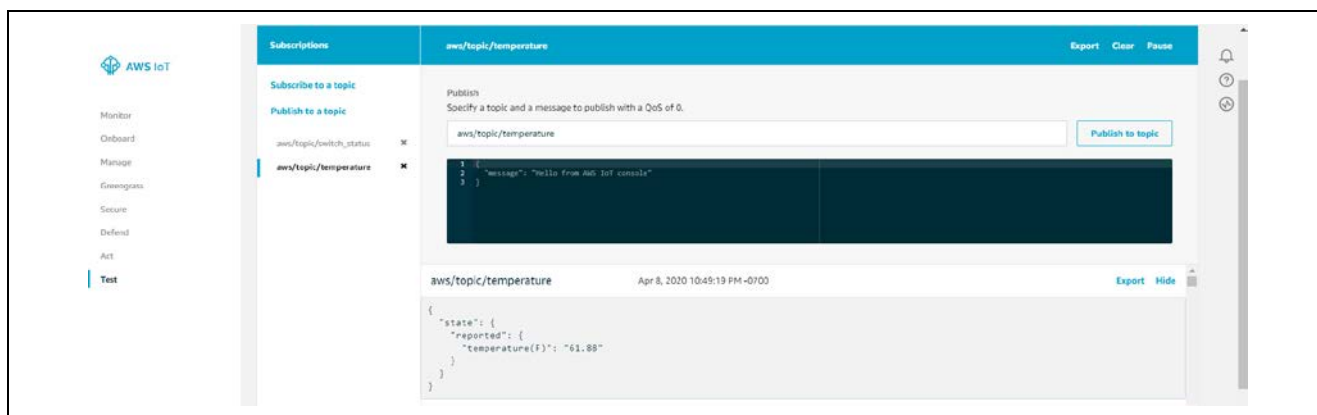


Figure 29. Subscribed Messages on the AWS IoT Screen

To Publish the message from Cloud to the Device, in order to turn ON/OFF the LEDs, select **PUBLISH to a Topic** and type the specific topic (“aws/topic/led”) and message in the respective window and click the **Publish to a Topic** as shown in the Figure 30.

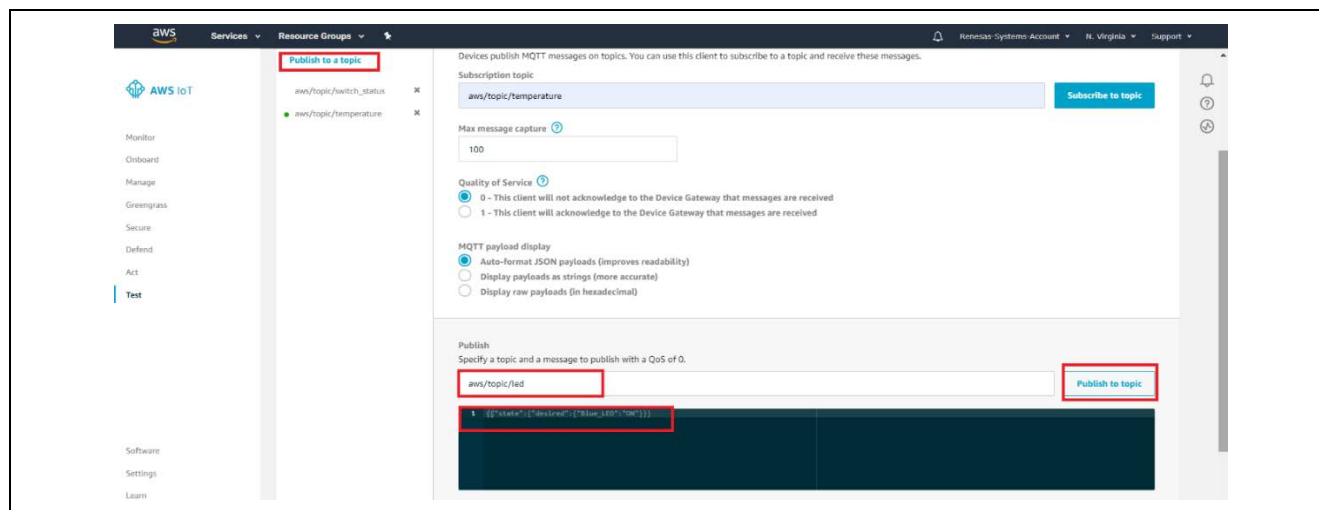
The actuation of the requested LEDs can be seen on the Demo Board. The application project allows control of the three user LEDS available on the board.

Note: The messages under Message Column are **case sensitive**; users need to take care of this while using them to turn the LEDs ON/OFF.

Only enter one message at a time. Copy the message ‘as-is’ and do not include any extra spaces.

Table 8. Messages for Toggling User LEDs

LED State	Message
RED LED ON	<code>{{"state":{"desired":{"Red_LED":"ON"}}}}</code>
RED LED OFF	<code>{{"state":{"desired":{"Red_LED":"OFF"}}}}</code>
BLUE LED ON	<code>{{"state":{"desired":{"Blue_LED":"ON"}}}}</code>
BLUE LED OFF	<code>{{"state":{"desired":{"Blue_LED":"OFF"}}}}</code>
GREEN LED ON	<code>{{"state":{"desired":{"Green_LED":"ON"}}}}</code>
GREEN LED OFF	<code>{{"state":{"desired":{"Green_LED":"OFF"}}}}</code>

**Figure 30. Publishing the Messages from the AWS IoT Screen**

Note: The LEDs are also used for other user indications in this Application.

1. Green LED Blinks periodically to indicate the heartbeat of the system and healthy network connectivity.
2. Blue LED Toggles based on the MQTT Activity (Default: Temperature Data every 30 seconds). Here the Blue LED activity indicates the successful MQTT Messages are being exchanged.
3. Blinking RED LED indicates the MQTT message failure activity is seen. This could be Message Publish or Subscribe failure.
4. Solid RED LED indicates the Initialization failed. The console Log will have the appropriate log for the error.

7. MQTT/TLS Module Next Steps

- For setting up a client using a device certificate signed by a preferred CA certificate, refer to the link: <https://docs.aws.amazon.com/iot/latest/developerguide/device-certs-your-own.html>
- For using a self-signed certificate to configure AWS, refer to the link: <https://developer.amazon.com/docs/custom-skills/configure-web-service-self-signed-certificate.html>

8. Cloud Connectivity References

- [1] International Telecommunication Union, "ITU-T Y.4000/Y.2060 (06/2012)," 15 06 2012. [Online]. Available: <http://handle.itu.int/11.1002/1000/11559>.
- [2] Amazon Web Services, "AWS IoT Core Features," [Online]. Available: <https://www.amazonaws.cn/en/iot-core/features/>.
- [3] Amazon Web Services, "AWS IoT Core," [Online]. Available: <https://www.amazonaws.cn/en/iot-core/>.
- [4] W. T. L. L. O. S. R. N. S. R. X. G. K. N. K. S. F. M. K. D. L. I. R. Valerie Lampkin, Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry, IBM Redbooks, 2012.

- [5] I. E. T. Force, "The Transport Layer Security (TLS) Protocol Version 1.2," [Online]. Available: <https://tools.ietf.org/html/rfc5246>.
- [6] Amazon Web Services, "AWS IoT Security," [Online]. Available: <https://docs.aws.amazon.com/iot/latest/developerguide/iot-security.html>.
- [7] Amazon Web Services, "Transport Security in AWS IoT," [Online]. Available: <https://docs.aws.amazon.com/iot/latest/developerguide/transport-security.html>.
- [8] International Telecommunication Union, "X.509 (10/19) Summary," 10 2019. [Online]. Available: https://www.itu.int/dms_pubrec/itu-t/rec/x/T-REC-X.509-201910-!!!SUM-HTML-E.htm.
- [9] Eclipse Foundation, "Eclipse Mosquitto™ - An open source MQTT broker," [Online]. Available: <https://mosquitto.org/>.
- [10] Amazon Web Services, "AWS IoT Device SDK C: MQTT," [Online]. Available: <https://docs.aws.amazon.com/freertos/latest/lib-ref/c-sdk/mqtt/index.html>.
- [11] R. Barry, "Mastering the FreeRTOS™ Real Time Kernel," in *A Hands-On Tutorial Guide*, 2016.
- [12] A. I. D. S. C. Documentation, "AWS IoT Device SDK C: MQTT Functions," [Online]. Available: https://docs.aws.amazon.com/freertos/latest/lib-ref/c-sdk/mqtt/mqtt_functions.html.
- [13] Amazon, "Configuring the FreeRTOS Demos," [Online]. Available: <https://docs.aws.amazon.com/freertos/latest/userguide/freertos-configure.html>.
- [14] "Amazon FreeRTOS mbedTLS," [Online]. Available: https://github.com/aws/amazon-freertos/blob/master/libraries/3rdparty/mbedtls/utls/mbedtls_utils.c.
- [15] Renesas Electronics Corporation, "Renesas Flash Programmer (Programming GUI) - Documentation," [Online]. Available: <https://www.renesas.com/us/en/products/software-tools/tools/programmer/renesas-flash-programmer-programming-gui.html#documents>.
- [16] Silex Technology, Inc., "SX-ULPGN-EVK - Evaluation kit for Ultra-Low-Power Hostless Wi-Fi IoT Module," [Online]. Available: <https://www.silextechnology.com/connectivity-solutions/embedded-wireless/sx-ulpkn-evk>.

Prerequisite : <https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html>. Please use this link to understand the AWS IoT and its configuration.

Website and Support

Visit the following vanity URLs to learn about key elements of the RA family, download components and related documentation, and get support.

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Apr.24.20	—	First release document
1.01	Jun.17.20	—	Updated for FSP 1.1.0

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.