Application Note

TPR0692A

GETTING STARTED WITH VAULTIC4XX TLS ON RASPBERRY PI

Table of Contents

1 Introduction

1.1 Overview

The VaultIC4xx secure microcontrollers are designed to bring a robust & unique digital identity to a device, essential for applications, such as:

- Creating a secure connection to a cloud or a local network using e.g. TLS
- Authenticating a USB-C device
- Authenticating a QI power transmitter

They provide the essential cryptographic functions and protect related private keys against tampering.

They are therefore particularly suited for securing TLS transactions.

The aim of this document is to explain how to setup and to use the VaultIC4xx TLS development kit to create a TLS transaction through a Raspberry Pi board.

After a quick presentation of the content of the kit, the next chapters will provide more information about the kit capabilities, how to install it and run the example projects.

To finish the appendixes will present how to use the AWS MQTTS demonstration, and provide the board schematic.

1.2 Content of the kit

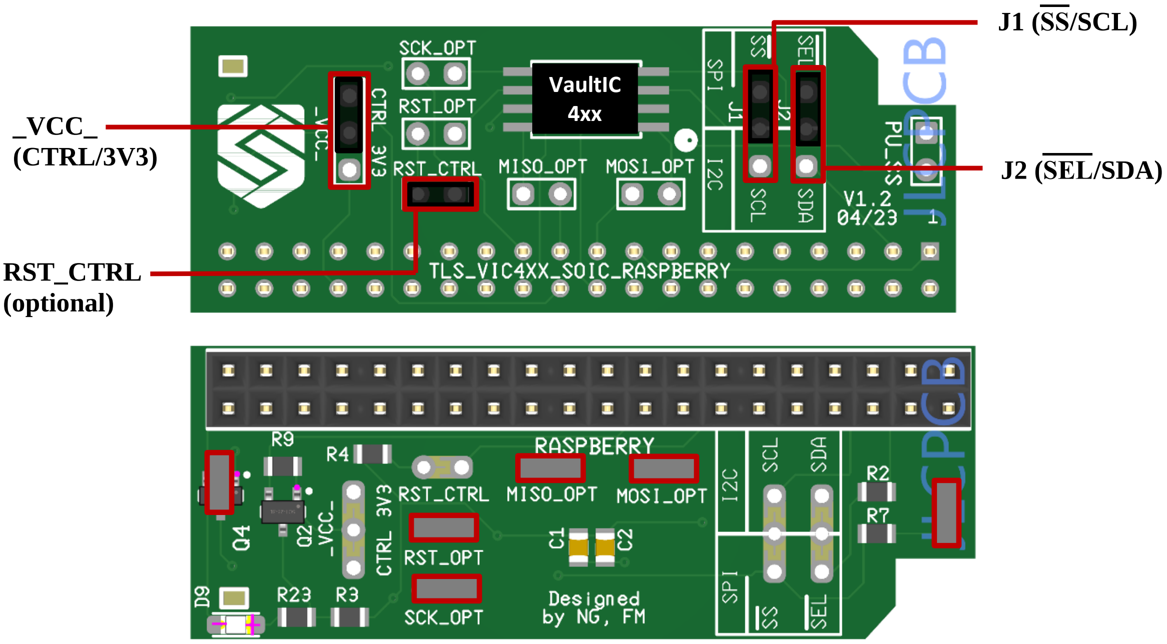
The development kit contains the following elements:

- a VaultIC4xx TLS evaluation board for Raspberry Pi
- a USB flash drive with all software and documentations required

2 Hardware presentation

The VaultIC4xx kit for raspberry includes one demo board to be plugged on the GPIO connector of a Raspberry Pi.

Figure 2-1 VaultIC4xx raspberry board with default configuration



The table below lists the available jumpers of the board and the default configuration when opening the kit:

Table 2-1 VaultIC4xx board jumpers

Name	Function	Default
J1 ($\overline{\text{SS}}$/SCL)	Choose signal connected to I²C SCL/$\overline{\text{SPI\_SS}}$ pin of the VaultIC4xx	Connected on $\overline{\text{SPI\_SS}}$ = SPI
J2 ($\overline{\text{SEL}}$/SDA)	Select I²C SDA line or force SPI selection $\overline{\text{SEL}}$ to 0	Connected on $\overline{\text{SEL}}$ = SPI
VCC	Select the power source of the board: - 3V3 : always on - CTRL: controlled by GPIO25 of raspberry	Connected on CTRL
RST_CTRL (optional)	When the jumper is inserted, the VaultIC4xx reset pin (not present on all devices) is controlled by the GPIO 27 of the Raspberry PI (connector pin 13). When the jumper is not inserted the reset is set to 1 with a 10kΩ pull-up.	Connected

Under the board, additional connections, listed below, can be done with soldered points:

Table 2-2 Additional connections

Name	Function	Default
LED	Connect or isolate a LED to indicate board power is on	Soldered
MOSI_OPT	Connect or isolate SPI MOSI line	Soldered
MISO_OPT	Connect or isolate SPI MISO line	Soldered
SCK_OPT	Connect or isolate SPI CLK line	Soldered
PU_SS	Connect or disconnect pull-up on SS	Soldered
RST_OPT	Connect or isolate the RST pin of the chip (not present on all devices)	Soldered

Note

The board allows to easily switch between SPI and I2C configurations just by moving J1 and J2 jumpers.

If always using the same configuration (I2C or respectively SPI), you can unsolder some of these points in case you need to use the other bus (SPI or respectively I2C).

Please refer to Appendix B -VaultIC4xx TLS Raspberry Pi board schematic to adapt your connections or contact your SEALSQ local FAE if you need help on this topic.

3 Software presentation

3.1 USB flash drive content

On the USB flash drive the software are zipped. In the zip file the directories organization is the following:

```
(root)
│
├ VaultIC-TLS
│ ├ certificates              ---->  files used to personalize the VaultIC secure element
│ │
│ ├ demos
│ │ ├ mbedtls                 ---->  demonstrations of VaulIC usage using mbedtls stack
│ │ │ ├ cert_req
│ │ │ ├ mqtt_aws
│ │ │ ├ tls_client
│ │ │ └ tls_server
│ │ │
│ │ ├ wolfssl                 ---->  demonstrations of VaulIC usage using wolfssl stack
│ │ │ ├ mqtt_aws
│ │ │ ├ tls_client
│ │ │ └ tls_server
│ │ │
│ │ └ config.cfg              ---->  used to switch between I2C or SPI
│ │
│ └ vaultic-tls
│   ├ vaultic_elib_4**        ---->  the VaultIC eLib (used to communicate with the device)
│   ├ vaultic_mbedtls         ---->  code to link mbedtls and vaultic
│   ├ vaultic_wolfssl         ---->  code to link wolfssl and vaultic
│   └ vaultic_tls-4xx
│     ├ apps
│     │ ├ check_tls_perso     ---->  program for reading certificates from the VaulIC device
│     │ ├ docs
│     │ └ perso_tls           ---->  program for loading certificates into the VaulIC device
│     └ src                   ---->  source code of the wrapper
│
├ CHANGES.txt
├ INSTALL.txt                 ---->  detailed installation instructions
└ TPR0692A_GettingStartedWithVault4xxTLS_Raspberry.pdf
```

3.2 Certificates

The VaultIC chip contained in the kit is not personalized yet, but an application included in the kit allows to personalize it through the Raspberry Pi with the data required for each demonstration.

The relevant certificate/key files in various format can be found in *“certificates”*.

File Name	Description
deviceCert.der	Client Certificate in DER format
deviceCert.pem	Client Certificate in PEM format
deviceKey.der	Client Key Pair in DER format
deviceKey.pem	Client Key Pair in PEM format
rootCA.key	Root CA Key Pair in PEM format
rootCACert.der	Root CA Certificate in DER format
rootCACert.pem	Root CA Certificate in PEM format
serverCert.der	Server Certificate in DER format
serverCert.pem	Server Certificate in PEM format
serverKey.der	Server Key Pair in DER format
serverKey.pem	Server Key Pair in PEM format
SFSRootCAG2.pem	AWS Root CA Certificate in PEM format

The key hierarchy is as follows:

- the rootCA is used to sign the device and server keys
- the device keys & certificates are used for the tls client
- the server keys & certificates are used for the tls server
- the SFSRootCA is used for the mbedtls aws client

4 Installation

4.1 Prerequisites

You will need the following software and account which installation is not described in the current kit:

- an AWS account: used with the AWS MQTTS demonstration
- openssl: used to generate certificates in the AWS MQTTS demonstration
  openssl is available with the raspberry Pi OS described below, if you want to use a Windows OS version of openssl you can obtain one here for example:
  https://slproweb.com/products/Win32OpenSSL.html

The steps required to install the kit are described in *“INSTALL.txt”*.

4.2 Raspberry Pi OS

The Raspberry Pi needs an OS to work. The VaultIC4xx TLS kit has been developed to run with the Raspberry Pi OS (previously called Raspbian).

All steps to install this OS are described on the official Raspberry Pi site:

https://www.raspberrypi.com/software/

Note

Recommended install configuration:

- Raspberry Pi FULL OS 32 bits
- Use advanced options (gear icon) to preconfigure all your Raspberry Pi (wifi etc)

To run all the examples, you will need the following software on the Raspberry Pi:

- git software and gcc (already present in the Raspberry Pi OS distribution)
- cmake: can be installed and checked with the following commands:

```
sudo apt update
sudo apt install -y cmake
cmake –version
```

4.3 Compiling and running example projects

Import and unzip the kit’s zip files **on the Raspberry Pi**.

All steps required to compile and run examples are described in the file *“INSTALL.txt”* located in the zip file.

Appendix A - AWS MQTTS demonstration

A.1 Overview

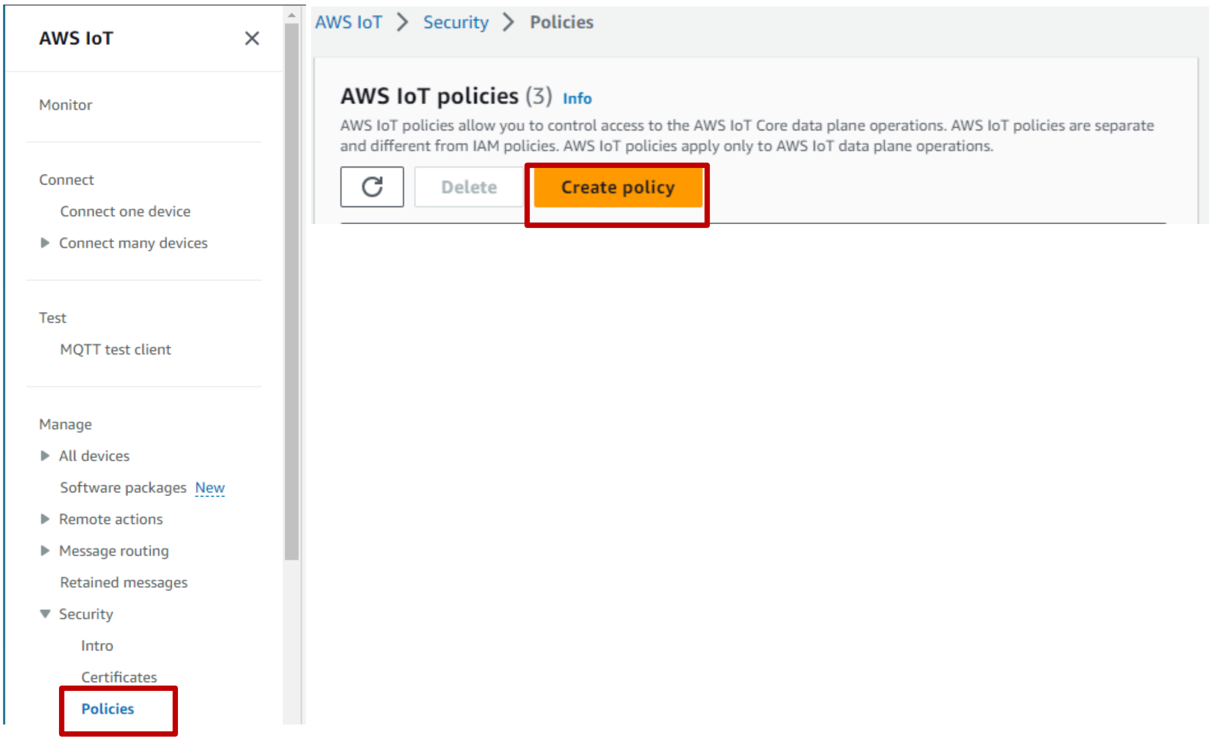
The following paragraphs detail the steps required to configure AWS, and how to use the demonstration.

Caution

The configuration provided here is intended for demonstration purpose, and might have to be adapted according to the configuration of your own AWS server.

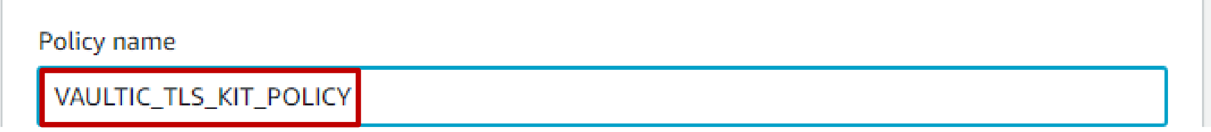
In the AWS portal go in the “IoT Core” service (you can use the search bar to easily find it).

A.2 Create policy



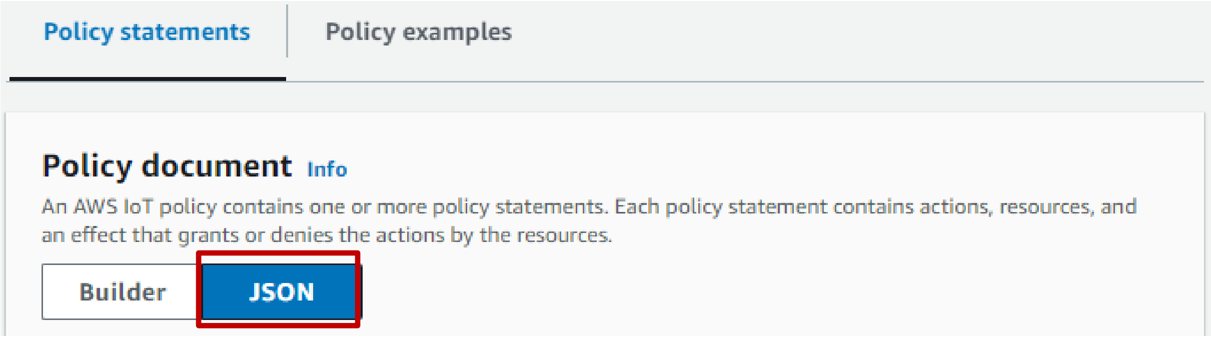
AWS IoT policies allow you to control access to the AWS IoT Core data plane operations. AWS IoT policies are separate and different from IAM policies. AWS IoT policies apply only to AWS IoT data plane operations.

Enter the policy name (any value):

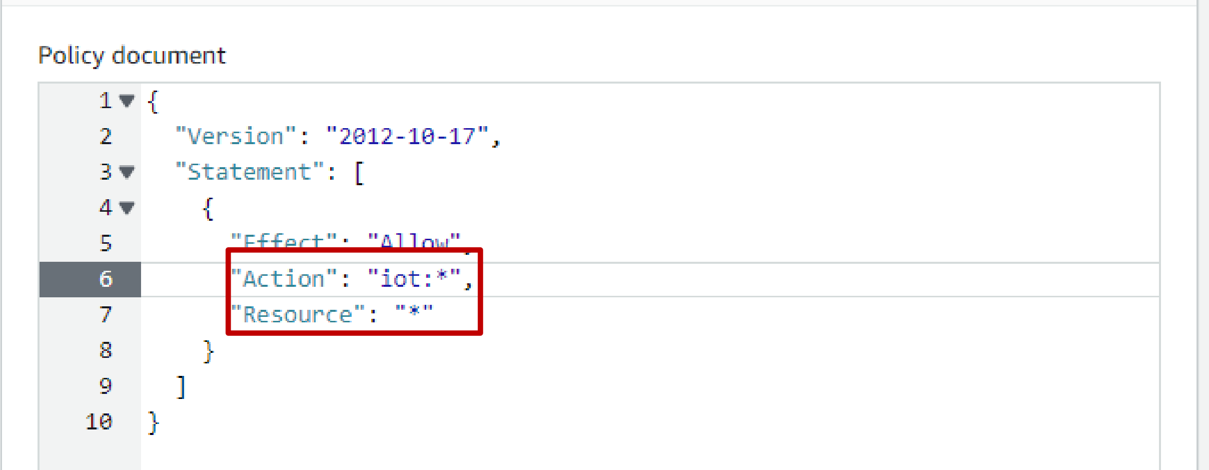


Policy name: VAULTIC_TLS_KIT_POLICY

Enter policy details:



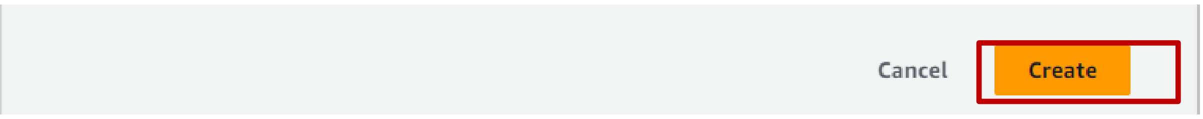
An AWS IoT policy contains one or more policy statements. Each policy statement contains actions, resources, and an effect that grants or denies the actions by the resources.


Policy document

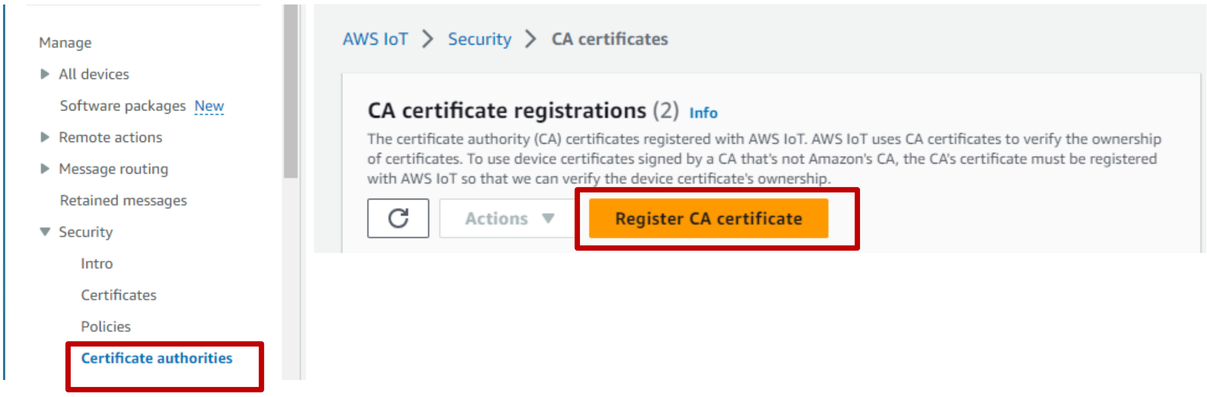
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:*",
      "Resource": "*"
    }
  ]
}
```

⚠ Caution

This policy is highly permissive, don’t use it in a production environment!



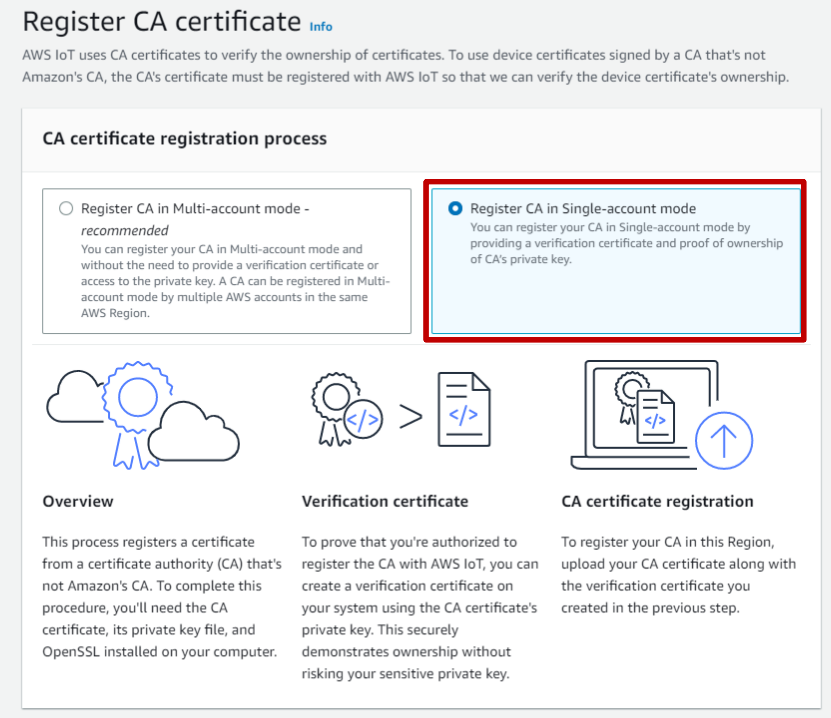
A.3 Register CA certificate



CA certificate registrations (2) Info

The certificate authority (CA) certificates registered with AWS IoT. AWS IoT uses CA certificates to verify the ownership of certificates. To use device certificates signed by a CA that's not Amazon's CA, the CA's certificate must be registered with AWS IoT so that we can verify the device certificate's ownership.

Chose the single account mode



Register CA certificate Info

AWS IoT uses CA certificates to verify the ownership of certificates. To use device certificates signed by a CA that's not Amazon's CA, the CA's certificate must be registered with AWS IoT so that we can verify the device certificate's ownership.

CA certificate registration process

○ Register CA in Multi-account mode - *recommended*
You can register your CA in Multi-account mode and without the need to provide a verification certificate or access to the private key. A CA can be registered in Multi-account mode by multiple AWS accounts in the same AWS Region.

● Register CA in Single-account mode
You can register your CA in Single-account mode by providing a verification certificate and proof of ownership of CA's private key.

Overview

This process registers a certificate from a certificate authority (CA) that's not Amazon's CA. To complete this procedure, you'll need the CA certificate, its private key file, and OpenSSL installed on your computer.

Verification certificate

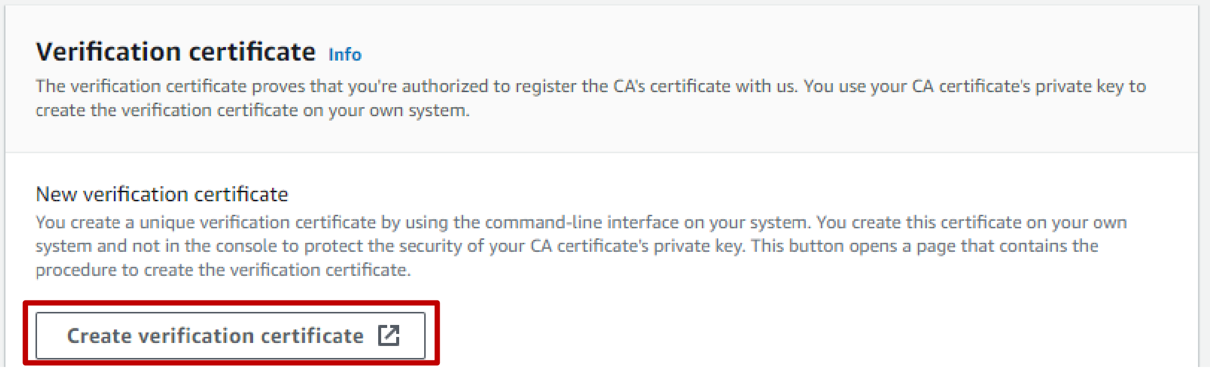
To prove that you're authorized to register the CA with AWS IoT, you can create a verification certificate on your system using the CA certificate's private key. This securely demonstrates ownership without risking your sensitive private key.

CA certificate registration

To register your CA in this Region, upload your CA certificate along with the verification certificate you created in the previous step.

and first create a verification certificate:

Note We recommend to right click on the “Create verification certificate” button and open link in a new tab (to easily come back to the Register CA Certificate page).



Verification certificate Info

The verification certificate proves that you're authorized to register the CA's certificate with us. You use your CA certificate's private key to create the verification certificate on your own system.

New verification certificate

You create a unique verification certificate by using the command-line interface on your system. You create this certificate on your own system and not in the console to protect the security of your CA certificate's private key. This button opens a page that contains the procedure to create the verification certificate.

You are redirected to this page:
[Create a CA verification certificate to register the CA certificate in the console – AWS IoT Core (amazon.com)]

Open a command prompt on your computer, and follow the different steps indicated.

Note These steps require openssl to be installed on your computer.

If using the default names of the AWS guide, run this command:

```
openssl genrsa -out verification_cert.key 2048
```

This will create a file *verification_cert.key* containing an RSA key pair.

Now run this command:

```
openssl req -new -key verification_cert.key -out verification_cert.csr
```

Note

The most important field here is the Common Name. As indicated in the AWS guide, other fields are optional here.

For some fields there will be a value proposed by default (and selected if you press Enter). If you enter '.', the field will be left blank.

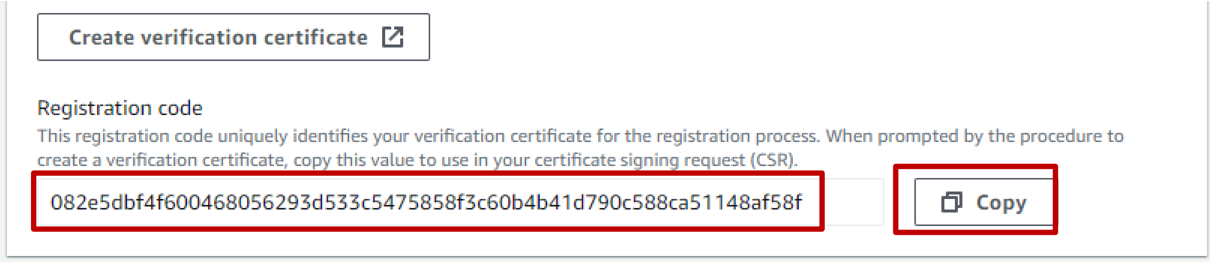
The first fields can be left empty:

```
Country Name (2 letter code) [AU]:.
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:
```

When asked for the Common Field:

```
Common Name (e.g. server FQDN or YOUR name) []:
```

Go back to the "Register CA Certificate page" and copy the Registration code displayed:



Registration code

This registration code uniquely identifies your verification certificate for the registration process. When prompted by the procedure to create a verification certificate, copy this value to use in your certificate signing request (CSR).

082e5dbf4f600468056293d533c5475858f3c60b4b41d790c588ca51148af58f

Paste it in the command prompt:

```
Common Name (e.g. server FQDN or YOUR name)
[]:082e5dbf4f600468056293d533c5475858f3c60b4b41d790c588ca51148af58f
```

Remaining information can be left blank, for example:

```
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

A new file *verification_cert.csr* is created, containing the verification certificate signing request.

Now create the verification certificate with this command:

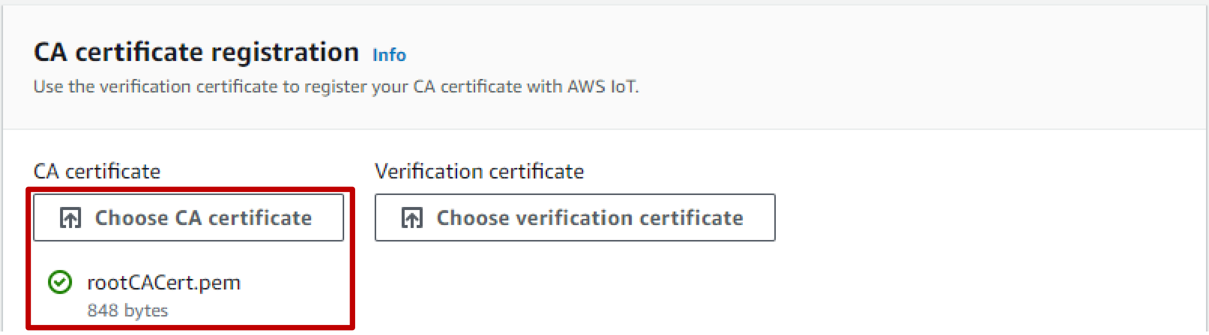
Note

You need the *rootCACert.pem* and *rootCA.key* files located in the *“certificates”* directory of the kit. Please refer to 3.2 Certificates for details.

```
openssl x509 -req -in verification_cert.csr -CA rootCACert.pem -CAkey rootCA.key -
CAcreateserial -out verification_cert.pem -days 500 -sha256
```

A new file *verification_cert.pem* is created, containing the verification certificate.

Now upload the CA certificate *rootCACert.pem*:

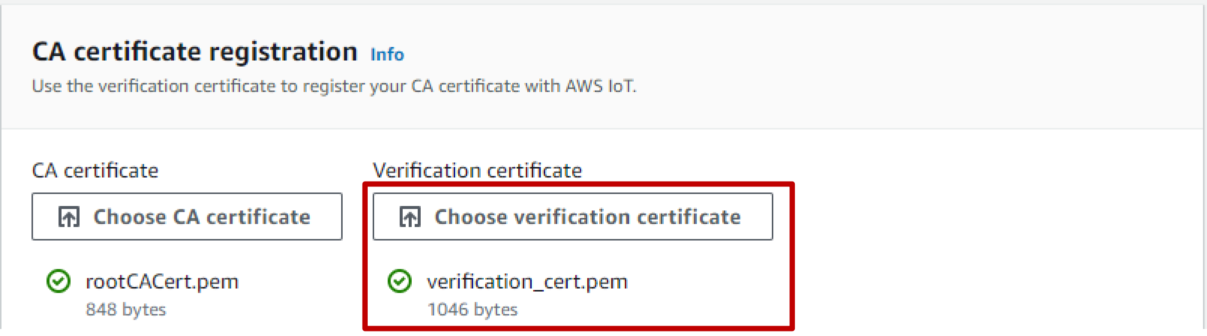


CA certificate registration Info

Use the verification certificate to register your CA certificate with AWS IoT.

rootCACert.pem
848 bytes

Upload the verification certificate *verification_cert.pem* created above:



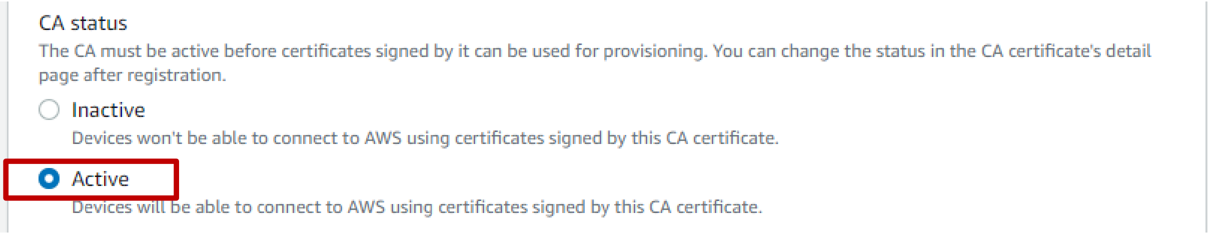
CA certificate registration Info

Use the verification certificate to register your CA certificate with AWS IoT.

rootCACert.pem
848 bytes

verification_cert.pem
1046 bytes

Activate the CA certificate:



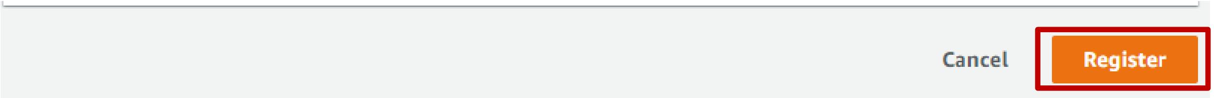
CA status

The CA must be active before certificates signed by it can be used for provisioning. You can change the status in the CA certificate's detail page after registration.

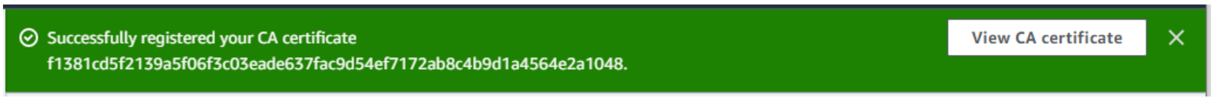
○ Inactive
Devices won't be able to connect to AWS using certificates signed by this CA certificate.

● Active
Devices will be able to connect to AWS using certificates signed by this CA certificate.

Finish the CA registration:

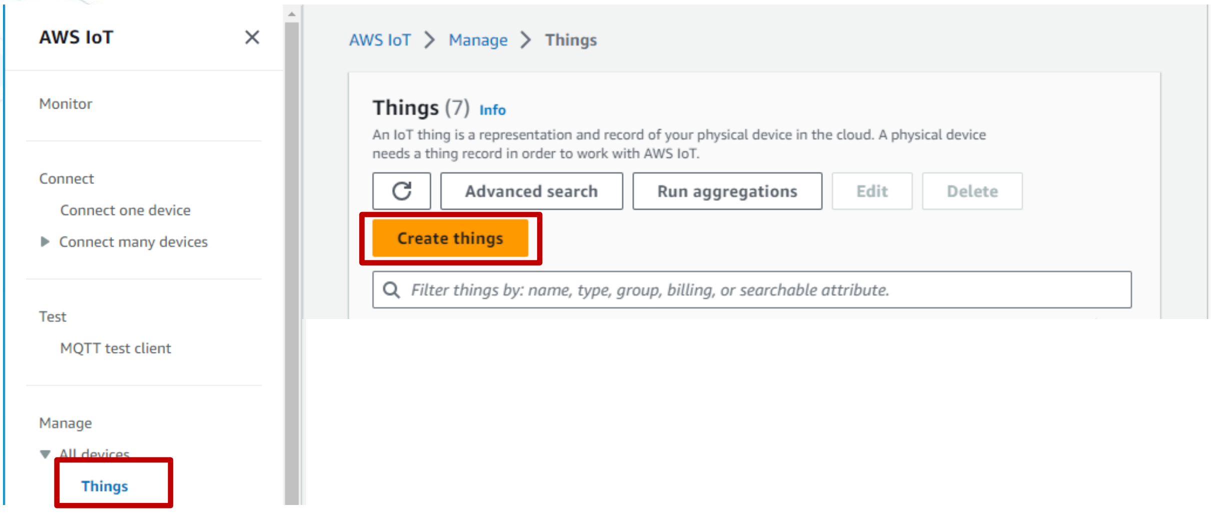


The CA configuration is now complete:

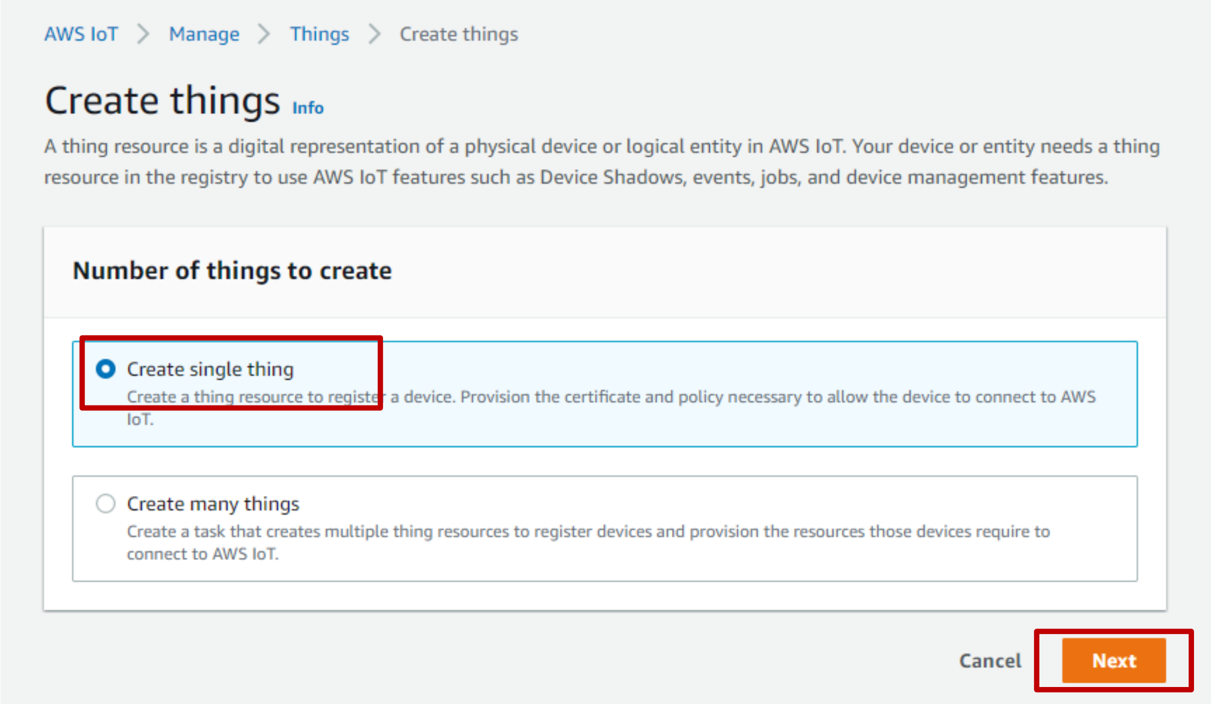


Successfully registered your CA certificate f1381cd5f2139a5f06f3c03eade637fac9d54ef7172ab8c4b9d1a4564e2a1048.

A.4 Create thing



An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

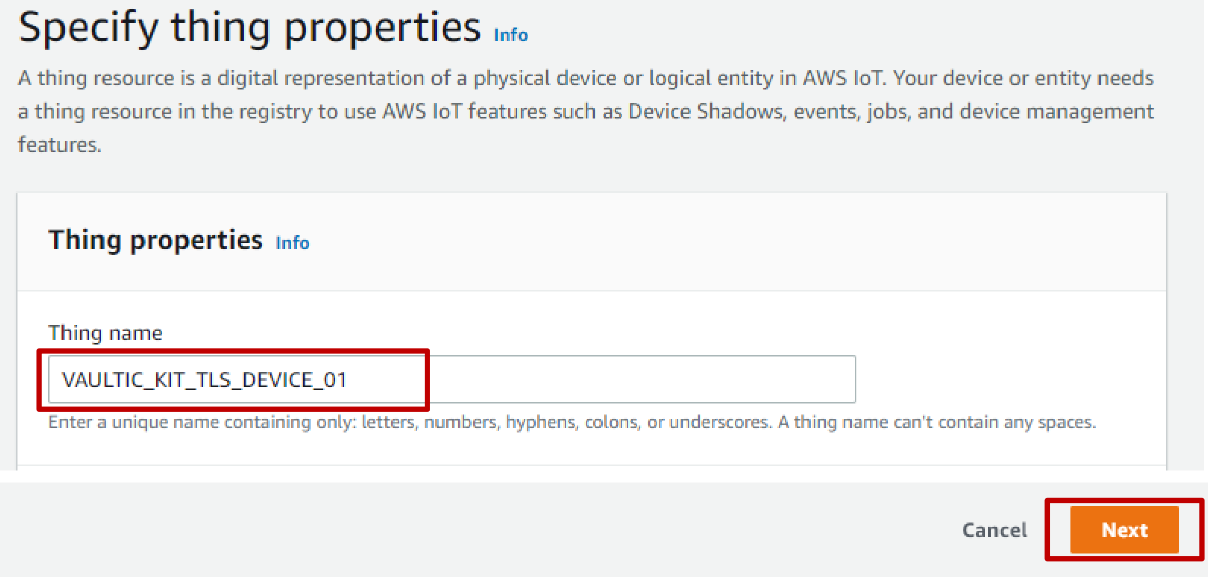

AWS IoT > Manage > Things > Create things
Create things Info
Number of things to create
Cancel
Next

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

● Create single thing
Create a thing resource to register a device. Provision the certificate and policy necessary to allow the device to connect to AWS IoT.

○ Create many things
Create a task that creates multiple thing resources to register devices and provision the resources those devices require to connect to AWS IoT.

Enter the thing name (any value):



Specify thing properties Info

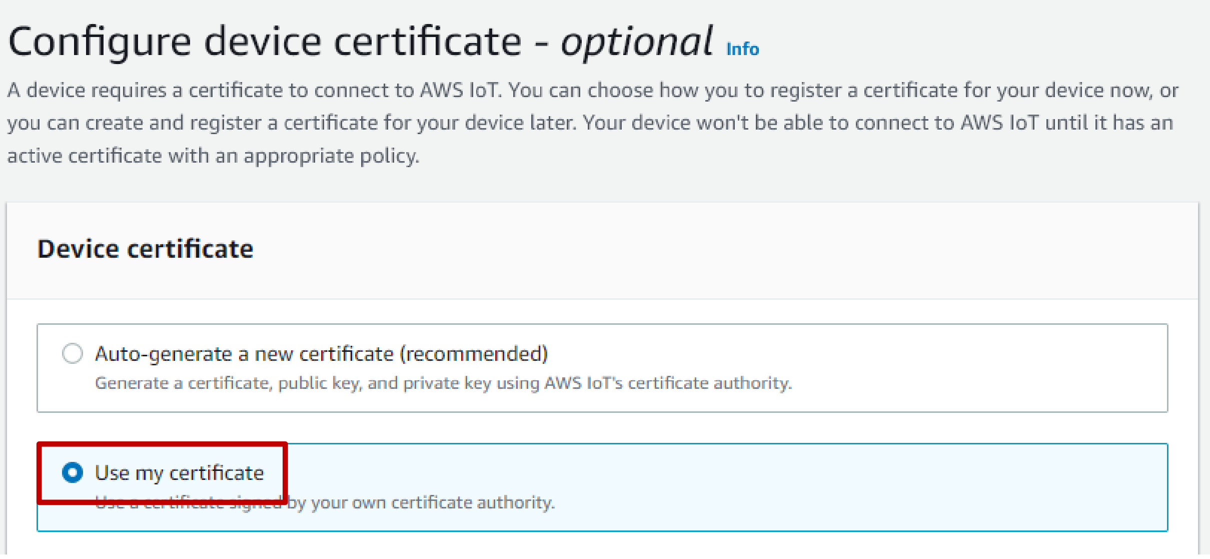
A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

Thing properties Info

Thing name: VAULTIC_KIT_TLS_DEVICE_01

Enter a unique name containing only: letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Attach the device certificate:



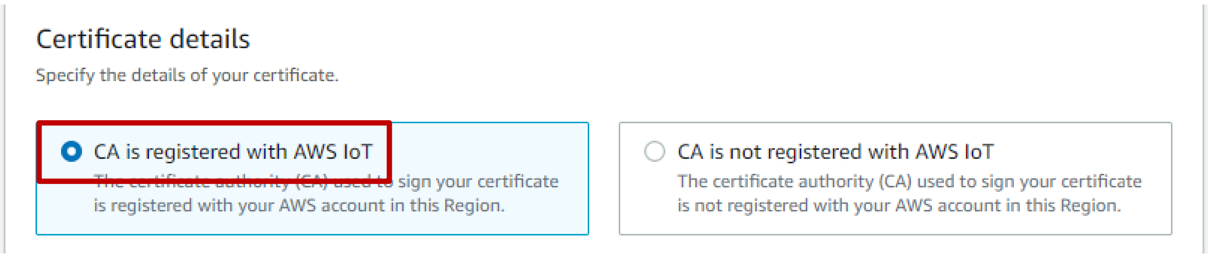
Configure device certificate - *optional* Info

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

Device certificate

○ Auto-generate a new certificate (recommended)
Generate a certificate, public key, and private key using AWS IoT's certificate authority.

● Use my certificate
Use a certificate signed by your own certificate authority.



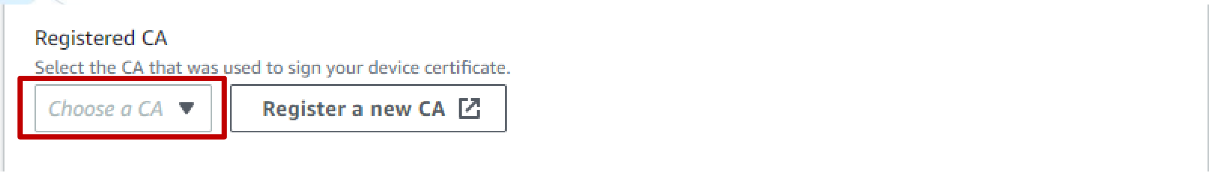
Certificate details

Specify the details of your certificate.

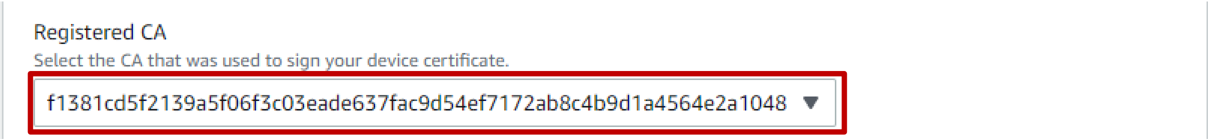
● CA is registered with AWS IoT
The certificate authority (CA) used to sign your certificate is registered with your AWS account in this Region.

○ CA is not registered with AWS IoT
The certificate authority (CA) used to sign your certificate is not registered with your AWS account in this Region.

Select the CA registered in A.3 Register CA certificate :


Registered CA

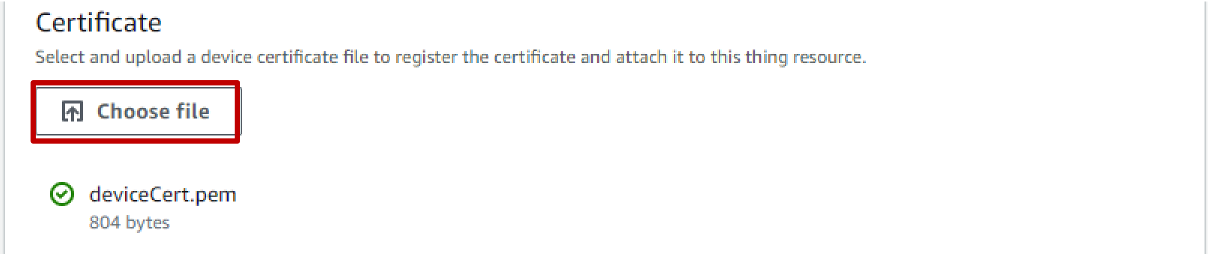
Select the CA that was used to sign your device certificate.


Registered CA

Select the CA that was used to sign your device certificate.

f1381cd5f2139a5f06f3c03eade637fac9d54ef7172ab8c4b9d1a4564e2a1048 ▼

Select the device certificate


Certificate

Select and upload a device certificate file to register the certificate and attach it to this thing resource.

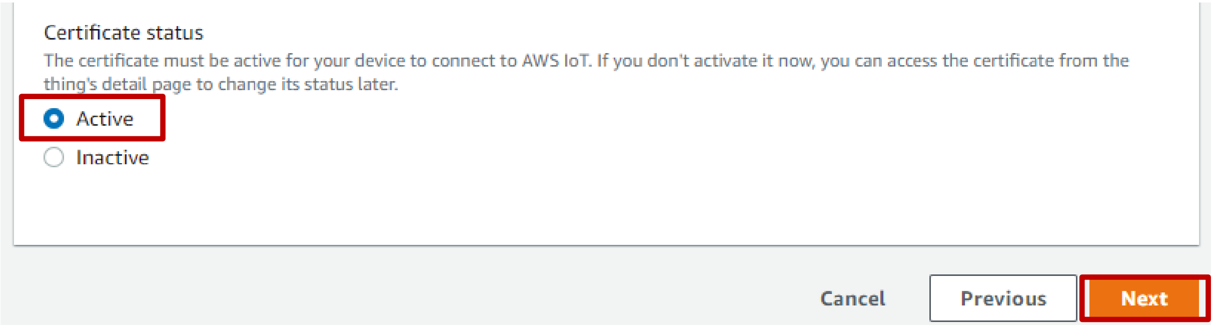
✓ deviceCert.pem

804 bytes

Note

You need the *deviceCert.pem* file located in the *"certificates"* directory of the kit. Please refer to 3.2 Certificates for details.

Check the certificate is active:

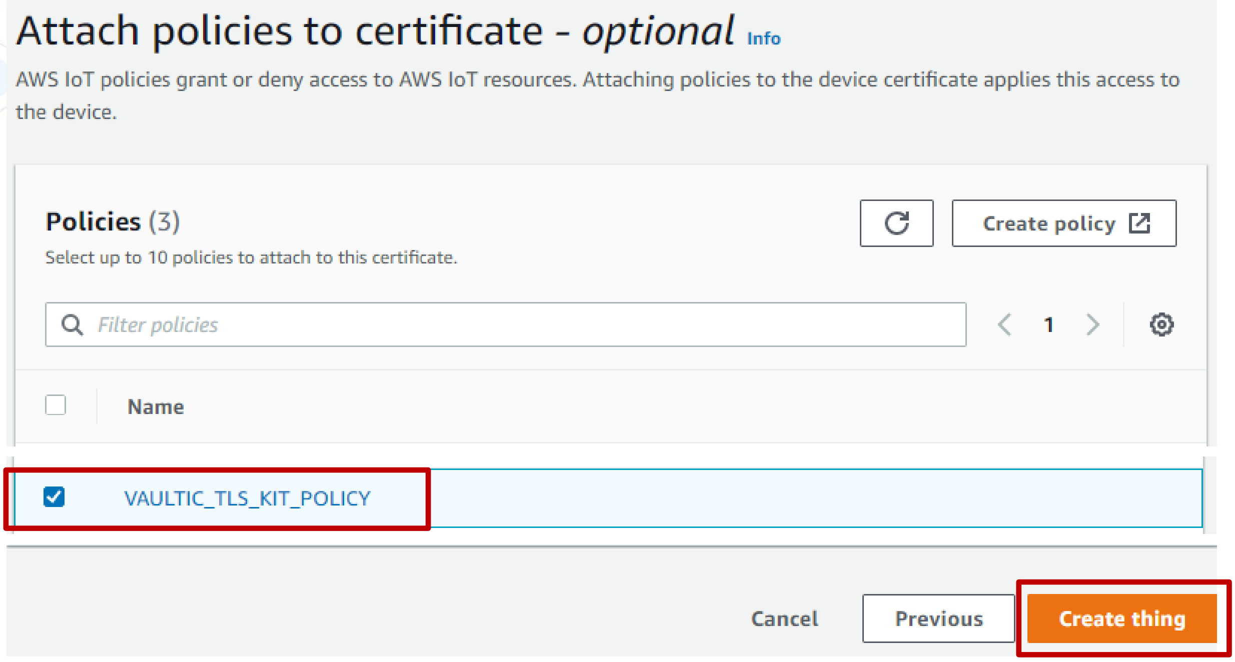

Certificate status

The certificate must be active for your device to connect to AWS IoT. If you don't activate it now, you can access the certificate from the thing's detail page to change its status later.

● Active

○ Inactive

Attach the policy created in A.2 Create policy :



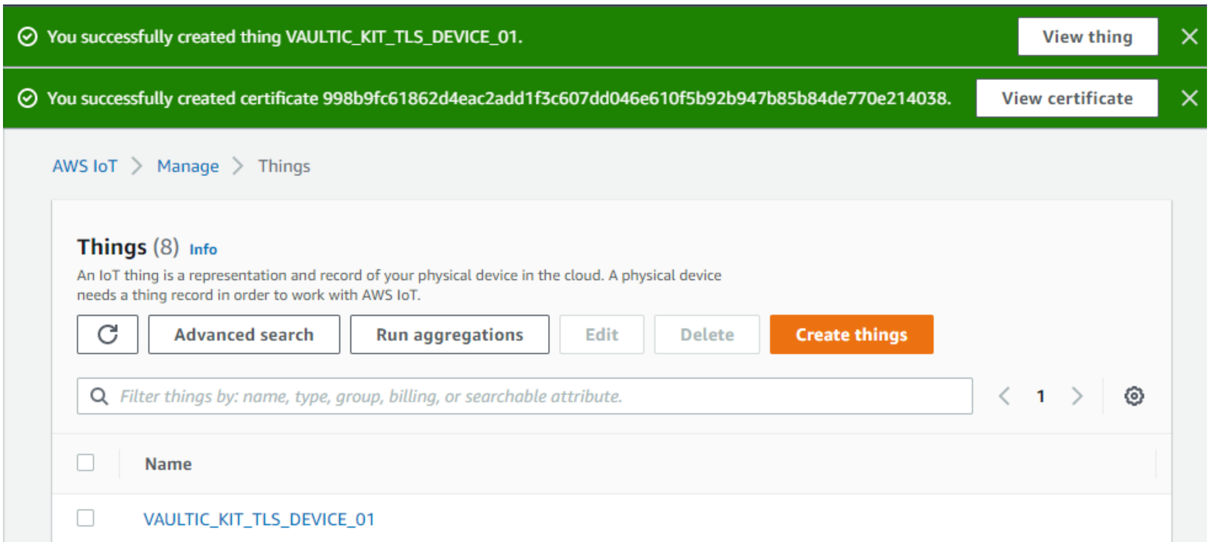
Attach policies to certificate - *optional* Info

AWS IoT policies grant or deny access to AWS IoT resources. Attaching policies to the device certificate applies this access to the device.

Policies (3)

Select up to 10 policies to attach to this certificate.

The configuration is now complete:



You successfully created thing VAULTIC_KIT_TLS_DEVICE_01.

You successfully created certificate 998b9fc61862d4eac2add1f3c607dd046e610f5b92b947b85b84de770e214038.

AWS IoT > Manage > Things

Things (8) Info

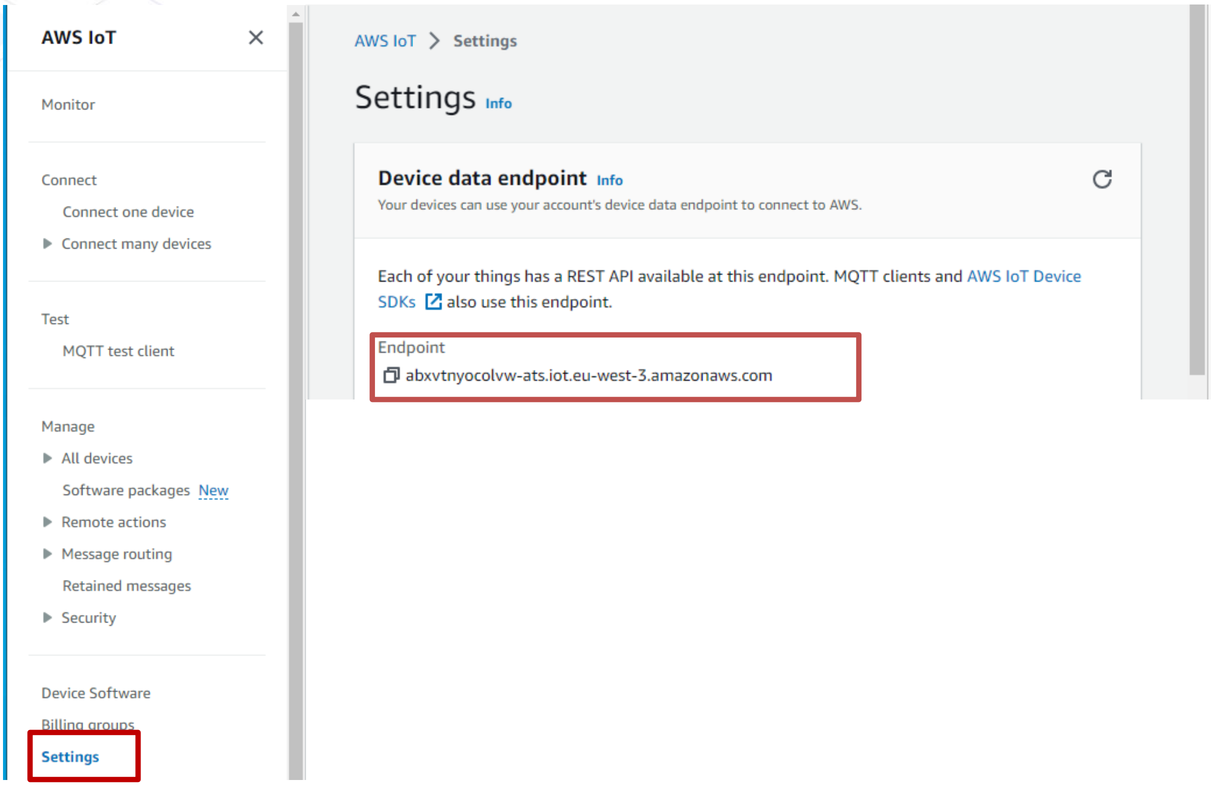
An IoT thing is a representation and record of your physical device in the cloud. A physical device needs a thing record in order to work with AWS IoT.

View thing
View certificate
Advanced search
Run aggregations
Edit
Delete
Create things
Filter things by: name, type, group, billing, or searchable attribute.
1
Name
VAULTIC_KIT_TLS_DEVICE_01

The thing is now ready to be used.

A.5 AWS Raspberry Pi client configuration

Retrieve the address of your AWS broker:



Your devices can use your account's device data endpoint to connect to AWS.

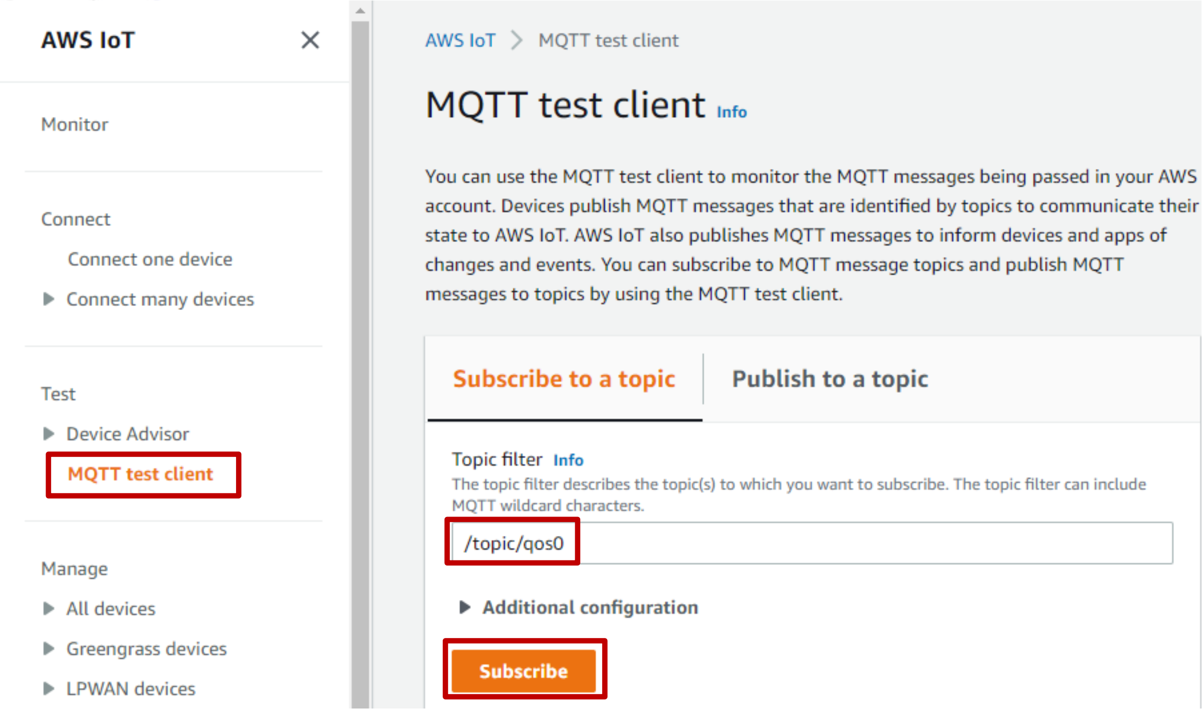
Each of your things has a REST API available at this endpoint. MQTT clients and AWS IoT Device SDKs also use this endpoint.

Endpoint
abxvtnyocolvw-ats.iot.eu-west-3.amazonaws.com

Now follow instructions in the README.txt file of the mqtt_aws demonstrations (located in demos/wolfssl/mqtt_aws and demos/mbedtls/mqtt_aws on the Raspberry Pi) to import the endpoint name in the mqtt client used to communicate with AWS.

A.6 Testing

Subscribe to */topic/qos0* to receive the messages sent by the Raspberry Pi demo:



AWS IoT > MQTT test client

MQTT test client Info

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

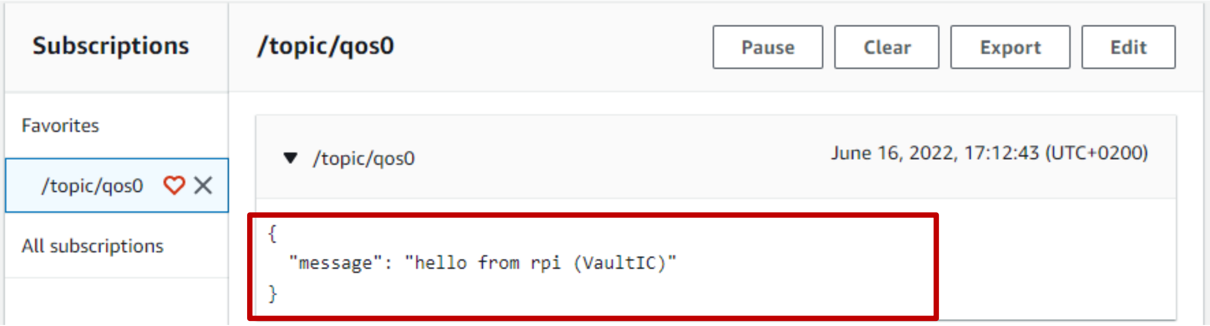
Topic filter Info

The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.

/topic/qos0

Run the **mqtt_aws** demo on the Raspberry Pi (as explained in “*README.txt*”).

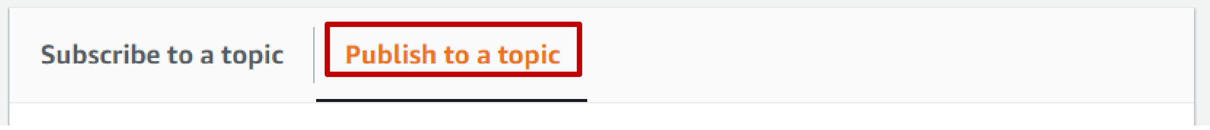
The message {"message": "hello from rpi (VaultIC)"} is immediately sent to AWS, and will appear here on the AWS side:



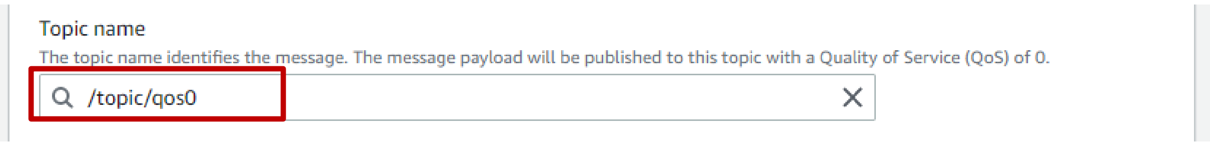
/topic/qos0 June 16, 2022, 17:12:43 (UTC+0200)

```
{
  "message": "hello from rpi (VaultIC)"
}
```

You can also send messages to the mqtt_aws demo:



Enter the same topic name than above: */topic/qos0*

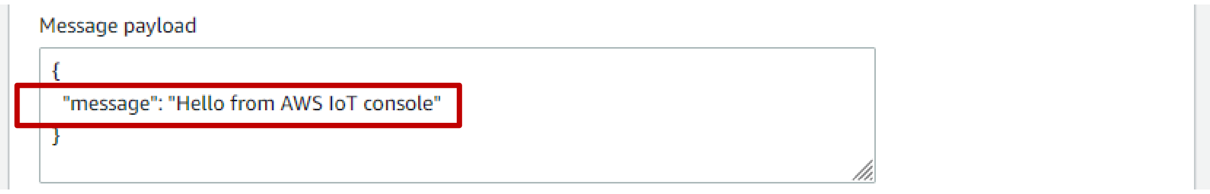


Topic name

The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.

/topic/qos0

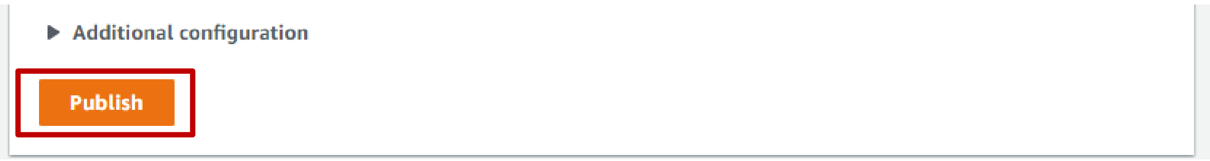
Enter any message for the mqtt_aws demo client



Message payload

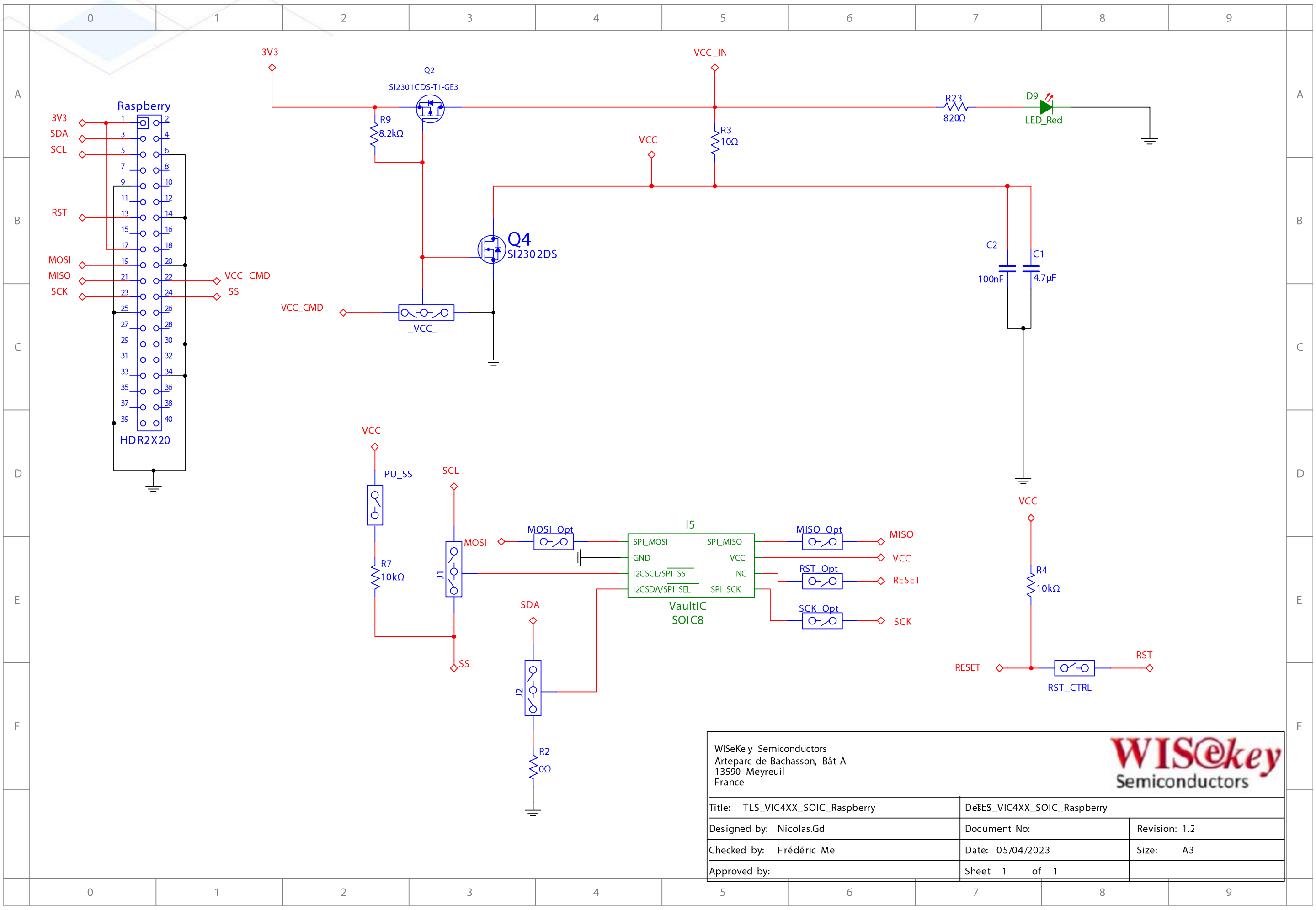
```
{
  "message": "Hello from AWS IoT console"
}
```

Click on Publish to send the message:



The message received from AWS will appear in the log console of the Raspberry Pi.

Appendix B - VaultIC4xx TLS Raspberry Pi board schematic



Reference list

[R1]	VaultIC4xx Technical Datasheet	TPR0684X

History

Version	Date	Comments
A	Aug 16, 2023	First Release

Headquarters

SEALSQ
Arteparc de Bachasson - Bat A
Rue de la Carrière de Bachasson
CS 70025
13590 Meyreuil - France
Tel: +33 (0)4-42-370-370
Fax: +33 (0)4-42-370-024

Product Contact

Web Site	***Technical Support***	***Sales Contact***
www.wisekey.com	dl_e-security@wisekey.com	sales@wisekey.com

Disclaimer: All products are sold subject to SEALSQ Terms & Conditions of Sale and the provisions of any agreements made between SEALSQ and the Customer. In ordering a product covered by this document the Customer agrees to be bound by those Terms & Conditions and agreements and nothing contained in this document constitutes or forms part of a contract (with the exception of the contents of this Notice). A copy of SEALSQ's Terms & Conditions of Sale is available on request. Export of any SEALSQ product outside of the EU may require an export Licence.

The information in this document is provided in connection with SEALSQ products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of SEALSQ products. EXCEPT AS SET FORTH IN SEALSQ'S TERMS AND CONDITIONS OF SALE, SEALSQ OR ITS SUPPLIERS OR LICENSORS ASSUME NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL SEALSQ BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, EXEMPLARY, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, LOSS OF REVENUE, BUSINESS INTERRUPTION, LOSS OF GOODWILL, OR LOSS OF INFORMATION OR DATA) NOTWITHSTANDING THE THEORY OF LIABILITY UNDER WHICH SAID DAMAGES ARE SOUGHT, INCLUDING BUT NOT LIMITED TO CONTRACT, TORT (INCLUDING NEGLIGENCE), PRODUCTS LIABILITY, STRICT LIABILITY, STATUTORY LIABILITY OR OTHERWISE, EVEN IF SEALSQ HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SEALSQ makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. SEALSQ does not make any commitment to update the information contained herein. SEALSQ advises its customers to obtain the latest version of device data sheets to verify, before placing orders, that the information being relied upon by the customer is current. SEALSQ products are not intended, authorized, or warranted for use as critical components in life support devices, systems or applications, unless a specific written agreement pertaining to such intended use is executed between the manufacturer and SEALSQ. Life support devices, systems or applications are devices, systems or applications that (a) are intended for surgical implant to the body or (b) support or sustain life, and which defect or failure to perform can be reasonably expected to result in an injury to the user. A critical component is any component of a life support device, system or application which failure to perform can be reasonably expected to cause the failure of the life support device, system or application, or to affect its safety or effectiveness.

The security of any system in which the product is used will depend on the system's security as a whole. Where security or cryptography features are mentioned in this document this refers to features which are intended to increase the security of the product under normal use and in normal circumstances.

The products identified and/or described herein may be protected by one or more of the patents and/or patent applications listed in related datasheets, such document being available on request under specific conditions. Additional patents or patent applications may also apply depending on geographic regions.