

## ANR032

ADRASTEIA-I AWS CLOUD  
CONNECTIVITY USING MQTT

VERSION 1.0

FEBRUARY 8, 2023

**WÜRTH ELEKTRONIK** MORE THAN YOU EXPECT

## Revision history

Document version	Notes	Date
1.0	<ul style="list-style-type: none"><li>Initial version</li></ul>	February 2023

# Contents

<b>1</b>	<b>IoT application</b>	<b>4</b>
1.1	Introduction . . . . .	4
1.2	The IoT stack . . . . .	4
1.3	IIoT application example . . . . .	6
1.4	System design using Adrastea-I and cloud platforms . . . . .	7
1.4.1	Embedded design . . . . .	8
1.4.2	Cloud platform design . . . . .	9
1.5	Building a prototype application . . . . .	10
<b>2</b>	<b>MQTT</b>	<b>11</b>
2.1	Publish/Subscribe . . . . .	11
2.2	Topics/Subscriptions . . . . .	11
2.3	QoS . . . . .	12
2.4	Message persistence . . . . .	12
2.5	Last Will and Testament . . . . .	12
2.6	Security . . . . .	12
2.7	MQTT on cloud platforms . . . . .	13
<b>3</b>	<b>AWS IoT Portal Setup and Configurations</b>	<b>14</b>
3.1	Create an AWS account . . . . .	14
3.2	Creating and Setting up Devices . . . . .	14
3.3	Configuration and Downloading Device Certificate and Key . . . . .	17
3.4	Setting Up and Attaching Policies to Device: . . . . .	19
3.5	Testing with Device: . . . . .	21
<b>4</b>	<b>Adrastea To Cloud Examples</b>	<b>22</b>
4.1	Adrastea-I to AWS Cloud Example . . . . .	22
<b>5</b>	<b>MQTT Related AT Commands</b>	<b>25</b>
5.1	MQTT AT Commands . . . . .	25
5.1.1	%MQTTCFG: Configure MQTT Connection Parameters . . . . .	25
5.1.2	%MQTTCMD: Communicate With MQTT Server (Broker) . . . . .	28
5.1.3	%MQTTEV: Notify About MQTT Events . . . . .	31
5.2	MQTT AT Commands for AWS . . . . .	34
5.2.1	%AWSIOTCFG: Configure AWS IoT Cloud . . . . .	34
5.2.2	%AWSIOTCMD: Communicate with AWS IoT Message Broker . . . . .	36
5.2.3	%AWSIOTEV: Notify About AWS IOT Events . . . . .	38
<b>6</b>	<b>Summary</b>	<b>40</b>
<b>7</b>	<b>References</b>	<b>41</b>
<b>8</b>	<b>Important notes</b>	<b>42</b>
8.1	General customer responsibility . . . . .	42
8.2	Customer responsibility related to specific, in particular safety-relevant applications . . . . .	42
8.3	Best care and attention . . . . .	42

8.4	Customer support for product specifications . . . . .	42
8.5	Product improvements . . . . .	43
8.6	Product life cycle . . . . .	43
8.7	Property rights . . . . .	43
8.8	General terms and conditions . . . . .	43
<b>9</b>	<b>Legal notice</b>	<b>44</b>
9.1	Exclusion of liability . . . . .	44
9.2	Suitability in customer applications . . . . .	44
9.3	Trademarks . . . . .	44
9.4	Usage restriction . . . . .	44
<b>10</b>	<b>License terms</b>	<b>46</b>
10.1	Limited license . . . . .	46
10.2	Usage and obligations . . . . .	46
10.3	Ownership . . . . .	47
10.4	Firmware update(s) . . . . .	47
10.5	Disclaimer of warranty . . . . .	47
10.6	Limitation of liability . . . . .	48
10.7	Applicable law and jurisdiction . . . . .	48
10.8	Severability clause . . . . .	48
10.9	Miscellaneous . . . . .	48

# 1 IoT application

## 1.1 Introduction

The Internet of Things (IoT) can be broadly defined as an umbrella term for a range of technologies that enable devices to connect and interact with each other. Interacting devices generating data provide the foundation for a range of new applications. Industrial automation, healthcare, smart home, smart cities, smart grids and smart farming are some of the areas in which IoT provides substantial benefits.

Dubbed the "fourth industrial revolution" or Industry 4.0, the Industrial IoT (IIoT) is the digitization of industrial assets and processes that connects products, machines, services, location-s/sites to workers, managers, suppliers, and partners. Closer networking of the digital world with the physical world of machines helps achieve higher productivity, safety, efficiency and sustainability.

The core task of any IoT solution is to get data from the field to the cloud where analysis of the same generates the desired value addition for the application. This application note aims to propose an elegant solution to achieve this task based on Adrastea-I Cellular module.

This chapter begins by describing the parts of a typical IoT application. Further, an application scenario is discussed with an example. Finally, a sensor-to-cloud IoT solution is presented using tools from Würth Elektronik eiSos. Specifically, a step-by-step description is presented to get sensor data over LTE-M/NB-IoT into the most popularly used cloud platforms like Amazon AWS.

## 1.2 The IoT stack

Irrespective of the area of application, an end-to-end IoT solution consists of the following components (figure 1).

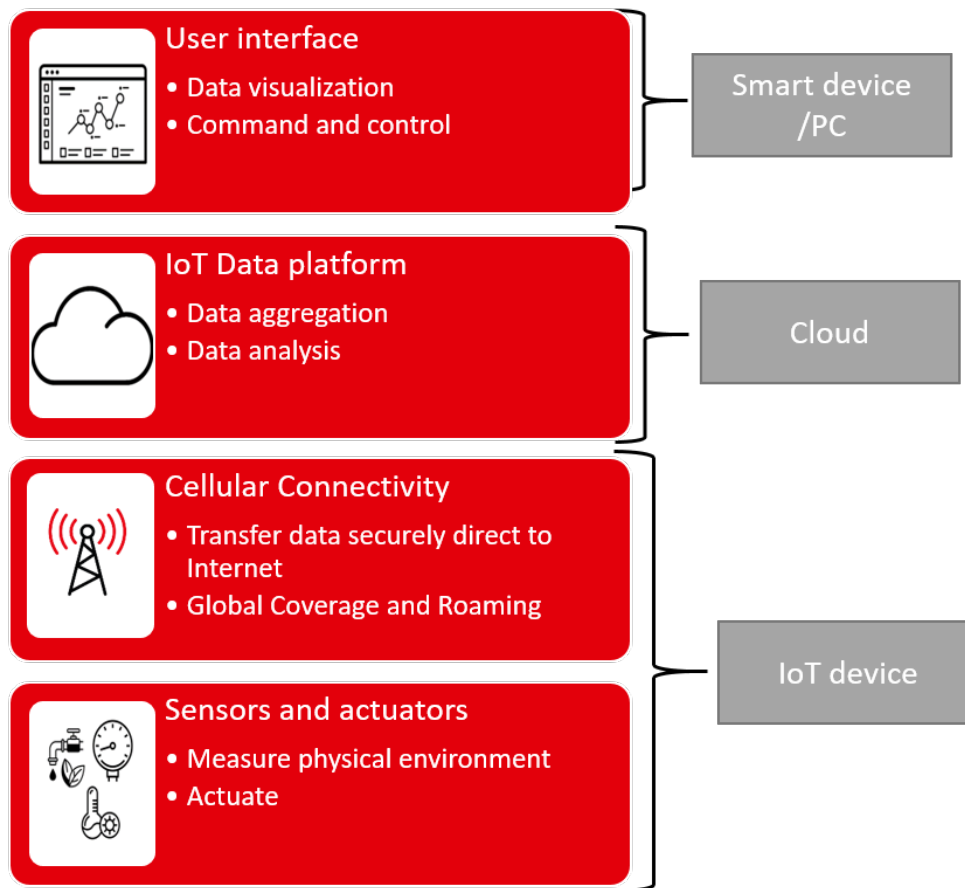


Figure 1: IoT application stack

- **Sensors and actuators:** This is a part of the system that directly interfaces with the physical environment. Sensors measure the state of the environment and interpret the same as analog or digital data. On the other hand, actuators activate a physical change in the measured environment. Advances in the field of electronics in general and semi-conductors in particular has led to availability of a wide range of sensors and actuators which are highly efficient and yet very compact.
- **Cellular connectivity:** Sensors and actuators are typically installed in devices with limited access to the digital world. Consider, for example, a temperature sensor mounted inside an industrial boiler. Cellular connectivity sends generated data direct to internet. This generated data is transmitted with security, authenticity and integrity in focus.

Modern embedded designs usually combine the above components into a single device (node) interacting with internet.

- **IoT Data platform:** This is the platform where the data is finally stored and presented for further analysis. Options here can range from a local database to a cloud server with redundancies. A typical platform consists of the components as shown in figure 2. The data platform enables the use of technologies like Artificial Intelligence (AI) and Machine Learning (ML) to perform advanced data analytics that generates value additions to the application.

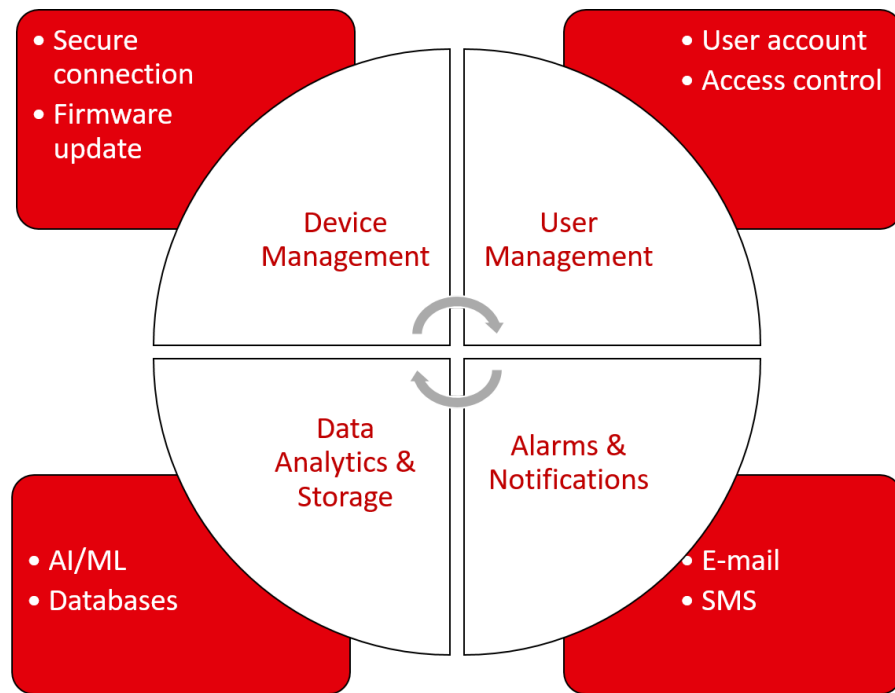


Figure 2: Components of a data platform

- **User interface:** This is the interface between the human users and the digital world. Here the status of the observed environment is presented in a human readable format. The user can take the necessary actions by interacting with this application

### 1.3 IIoT application example

The IIoT creates a universe of sensors that enables an accelerated deep learning of existing operations that allows rapid contextualization, automatic pattern, and trend detection. This leads to true quantitative capture of qualitative operations resulting in better quality, efficiency, higher safety, lower costs and several other benefits. This has led to the use of IoT in several application use cases.

**Remote monitoring and control** of production/storage environment (temperature, humidity, pressure etc.) is one of the essential tasks in the industry. Maintaining optimal conditions and automating this process has been a challenge. This use case is considered in this application note and an IoT based approach to solve this problem is presented here.

The above-mentioned task requires sensors/actuators to interact with the environment, cellular connectivity to transmit the data, a cloud platform to store data and a user interface to enable human interaction. The architecture of such a system is as shown in figure 3.

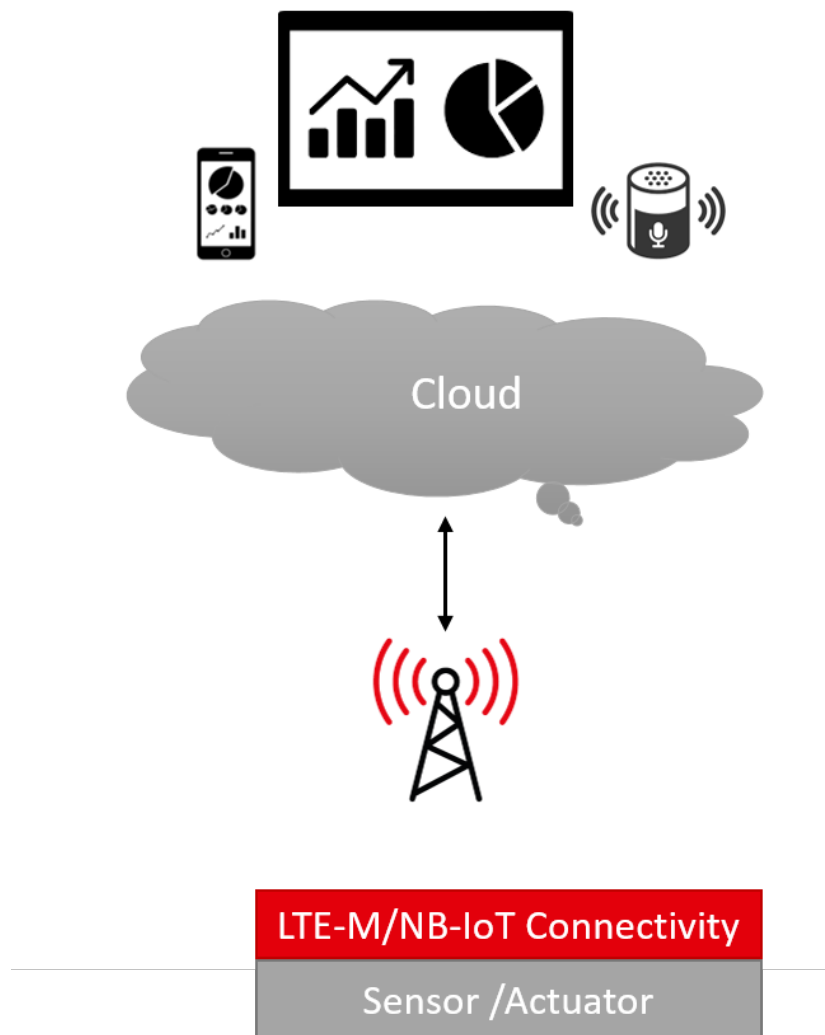


Figure 3: System architecture

## 1.4 System design using Adrastea-I and cloud platforms

Figure 4 illustrates the design that realizes the architecture described in 1.3. In this example, sensor data is transferred to the cloud platforms using the Cellular connectivity where it is stored and processed.



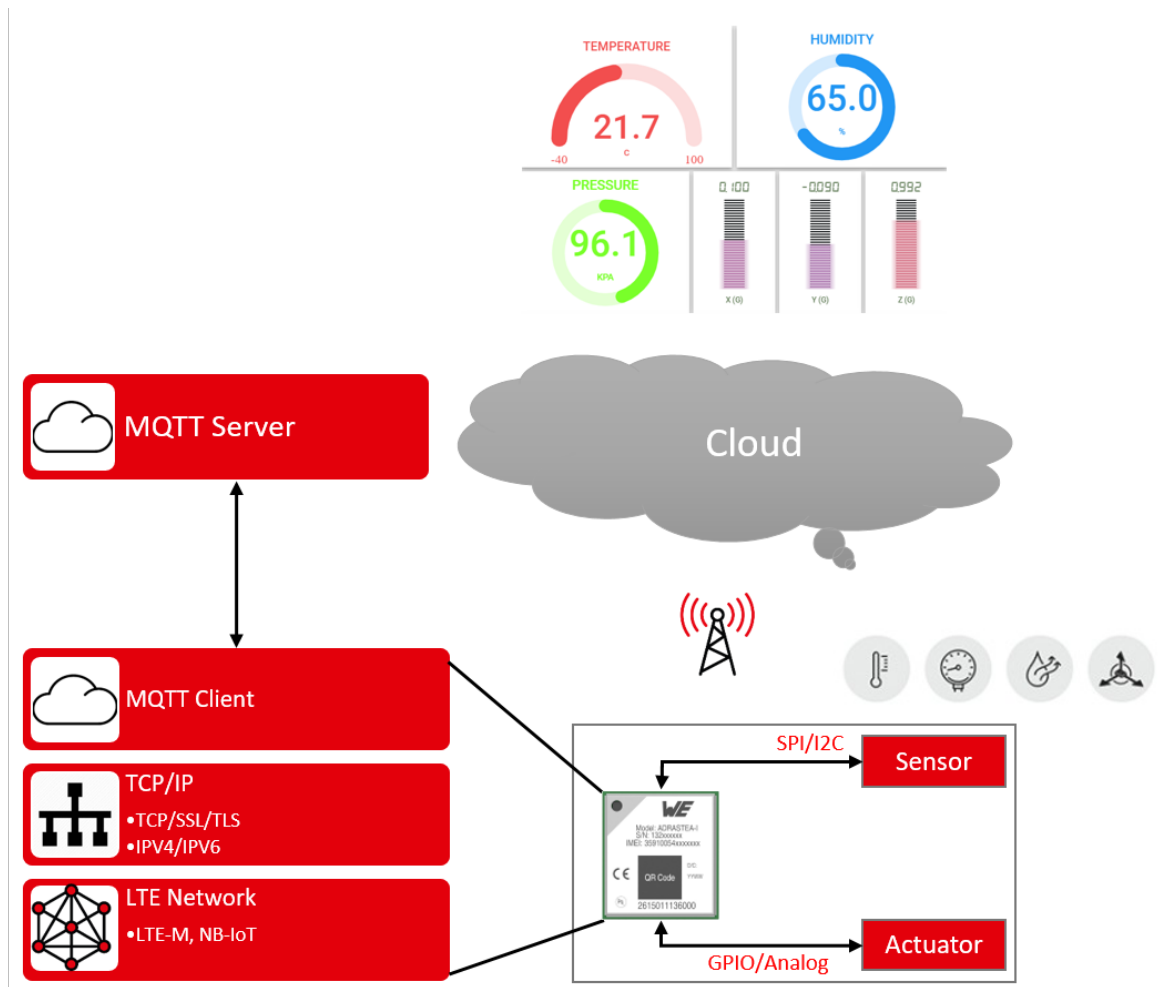


Figure 4: System design

### 1.4.1 Embedded design

The Adrastea-I is a LTE-M/NB-IoT Cellular module with integrated GNSS, integrated ARM Cortex-M4 and 1MB Flash memory for customer developed applications. With an on-board TCP/IP stack and MQTT protocol implemented out-of-the-box, Adrastea-I acts as the perfect building block for an IoT application.

Integrated ARM Cortex-M4 connects and controls sensors/actuators over standard interfaces like I2C, SPI, GPIO or analog. The module can be operated through one of two available cellular communication technologies:

- LTE-M or
- NB-IoT.

Further, the MQTT-client implemented on-board the Adrastea-I can be configured to connect to a broker/server running in the cloud. Most of the commonly used cloud platforms today use a type of secure MQTT protocol to connect devices and exchange data. Hence, Adrastea-I provides a direct communication link to the cloud without the need of a gateway device in between.

### 1.4.2 Cloud platform design

Hundreds of cloud platforms exist in the market today. In this example, the one of the most popular platforms, Amazon Web Services (AWS) is considered. Most cloud platforms are made up of cloud services, which offer specialized functionalities like,

- Device management
- User management
- Data storage
- Data streaming
- Data visualization
- AI/ML
- Security
- Networking
- Billing and cost management

These services act as building blocks for any IoT application. The challenge here is to pick and choose the right services and combine them into a secure, flexible and scalable application that serves the desired purpose.

In this application note, one such combination is considered keeping in mind the most frequently occurring use cases.

**Amazon AWS:** Figure 5 shows a sample application using Amazon AWS. In this example, the devices are provisioned to connect securely to the Amazon AWS IoT core. The incoming data is then processed using the AWS lambda function. Using the thresholds stored in the Dynamo DB, the lambda function detects the thresholds and sends notifications to the user using the notification service (Amazon SNS). Further, the data is streamed to an S3 bucket using the Kinesis Firehose streaming service. The stored data can be used for visualization using Amazon Quicksight service.

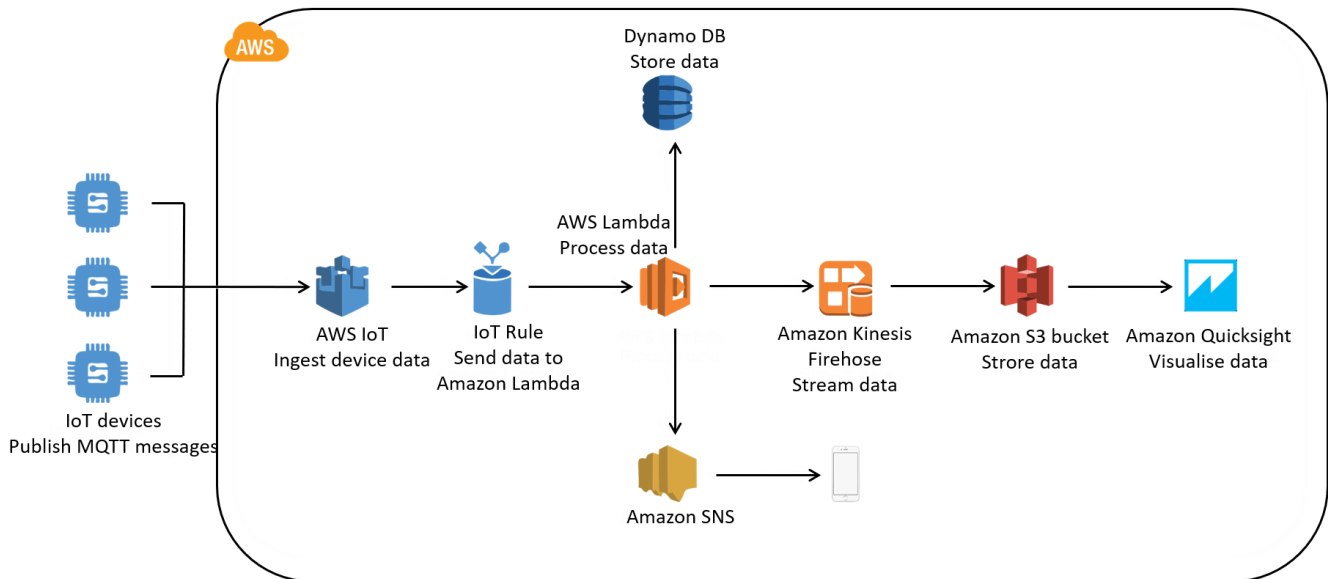


Figure 5: AWS example

## 1.5 Building a prototype application

Rest of this application note is intended to describe the process of building a prototype for an end-to-end IIoT solution (from sensor to cloud) using tools from Würth Elektronik eiSos. Chapter 2 gives a brief introduction to the MQTT protocol and chapter 4 provides a step-by-step description for building a prototype IoT application including the hardware, firmware and the cloud components.

## 2 MQTT

MQTT - Message Queuing Telemetry Transport is a lightweight application layer protocol based on a publish/subscribe messaging mechanism. This protocol was designed for resource constrained and unreliable networks with limited bandwidth and high latency. These characteristics make MQTT suitable for low-power, low-bandwidth IoT applications. Inherently, the MQTT protocol offers some degree of assurance of delivery thereby offering the robustness necessary for industrial machine-to-machine communication.

MQTT was originally developed by IBM and the version 3.1.1 is an OASIS standard that is open and royalty-free [3]. It is based on client-server architecture where every client connects to a server (broker) over TCP resulting in a star topology. Once connected, the clients send and receive messages using the publish/subscribe mechanism.

### 2.1 Publish/Subscribe

Data transfer in MQTT takes place based on publish/subscribe mechanism. The clients connected to the broker can publish messages under certain topics. The clients can also subscribe to topics that are of interest to them. When a client publishes a message on a topic, the broker forwards the message to any client that has subscribed to the topic. This mechanism enables bi-directional communication with an extremely low overhead (2-byte header).

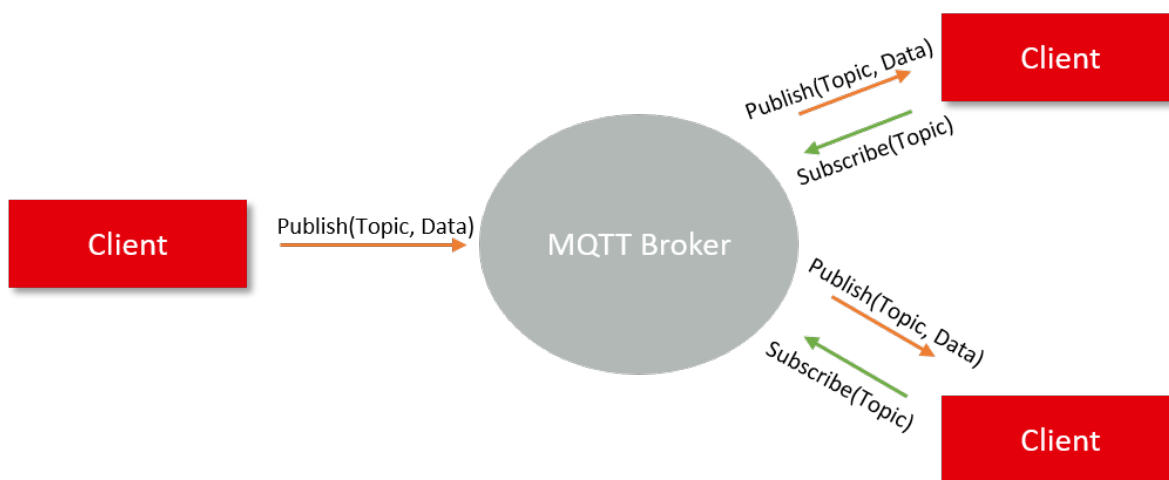


Figure 6: Publish/Subscribe mechanism

### 2.2 Topics/Subscriptions

Messages in MQTT are always published on topics. A hierarchy can be created in topics using the '/' character. For example, the status of smart light in the living room can have a topic "home/lighting/living\_room/ligth\_index".

Clients can create subscription on a topic by explicitly mentioning the topics or by using wildcard characters. There are two wildcards available, '+' and '#'. '+' matches any topics on a

single hierarchical level whereas '#' matches any number of levels. For example, subscribing to "home/lighting/+/light\_index" would result in getting status change messages of all lights with "light\_index" from all rooms of the house. On the other hand subscribing to "home/lighting/#" results in messages with all lights (all light indices) from all rooms.

This feature makes MQTT modular and highly scalable. Inserting a new node to an existing network does not require a lot of configuration.

## 2.3 QoS

Based on the requirement of the application one of the following levels of Quality-of-Service (QoS) can be chosen.

- **QoS level 0:** The broker/client delivers the message only once without acknowledgement of reception. The reliability in this case is same as that of the underlying TCP.
- **QoS level 1:** The broker/client delivers the message at least once. In this case an acknowledgement of received packet is done. This case however does not handle duplicate packets.
- **QoS level 2:** The broker/client delivers the message exactly once using a four-step handshake. This in turn offers higher reliability at the cost of lower throughput.

## 2.4 Message persistence

The publisher can specify if a message published to a topic has to be retained. If marked as retained, the broker retains the message and sends it to all new subscriptions. This acts as the "last known good" mechanism where nodes that come into network do not have to wait long to get the first message.

## 2.5 Last Will and Testament

This mechanism enables the client to publish one last message to all subscribed clients when it abruptly disconnects from the network. Clients can send a last will and testament message to the broker at any point. If the broker detects that a client has gone offline without a disconnect message, it sends the LWT message on the specified topic. This mechanism is very helpful to detect node failures in due to battery failure or networks outages.

## 2.6 Security

MQTT offers basic authentication where the client has to send a username and password with the connect message. The broker can authenticate the connection and allow or disallow a client. However, the user has to run MQTT over TLS/SSL to enable end-to-end encryption and advanced client authentication.

## **2.7 MQTT on cloud platforms**

Most cloud platforms support the MQTT protocol although with subtle variations. Both Amazon AWS and Microsoft Azure support only the secure version of MQTT with authentication and end-to-end encryption built in. Please refer to [1] and [2] for more details.

## 3 AWS IoT Portal Setup and Configurations

This chapter describes how to configure and use our Adrastea-I to communicate with AWS IoT portal. Before you start using it, read carefully the introduction in the AWS IoT Core website <https://aws.amazon.com/iot-core/>.

There are 5 major steps to set up your AWS IoT portal:

- Create an AWS account (Section: 3.1 )
- Setting Up Device Type and Creating Devices Section: 3.2
- Configuration and Downloading Device Certificate and Key Section: 3.3
- Setting Up and Attaching Policies to Device Section: 3.4
- Testing with Device Section: 3.5



Due to AWS's continuous evolution, some information provided in this document can change. Please refer to the links from AWS in each section to get the latest information.

### 3.1 Create an AWS account

To get started with the Amazon Web Services (AWS) IoT service, it is necessary to set up the AWS account and permissions. For details on how to create an AWS account, visit the AWS official website link:

<https://aws.amazon.com/premiumsupport/knowledge-center/create-and-activate-aws-account/>

Detailed instructions are available in sections Sign up for an AWS account and Create a user and grant permissions at:

<https://docs.aws.amazon.com/iot/latest/developerguide/setting-up.html>

### 3.2 Creating and Setting up Devices

1. Sign in to the AWS Management Console.
2. Search and Select "IoT Core" service in search bar.

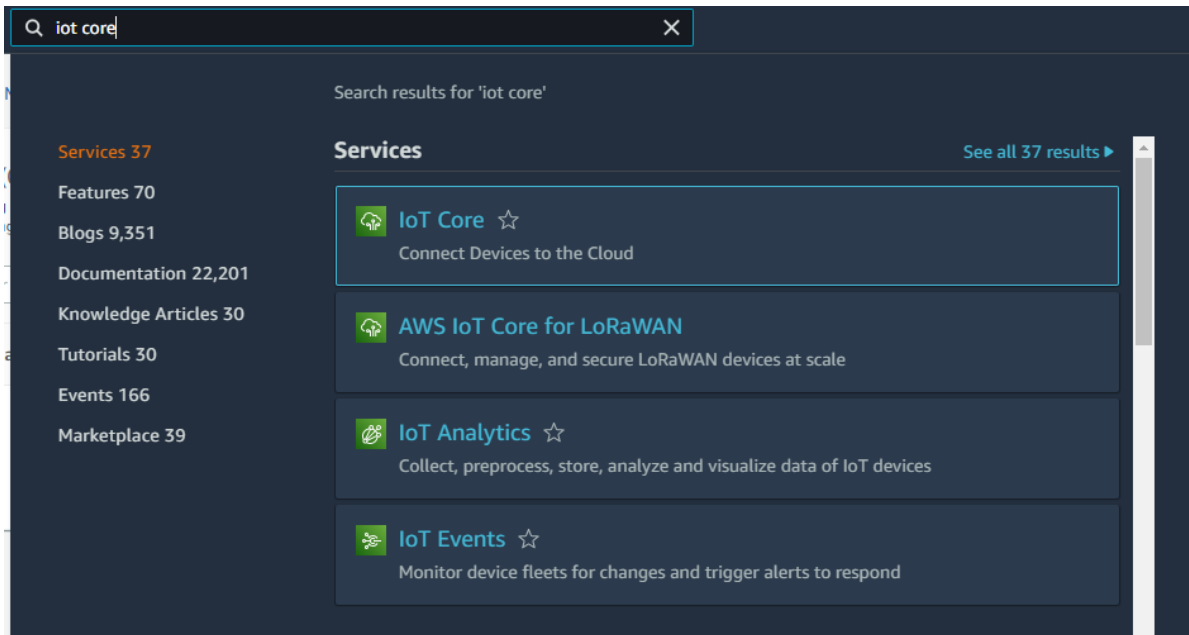


Figure 7: AWS IoT Core Microservice

3. Create a device type from AWS IoT -> Manage -> Things. Click "Create things".

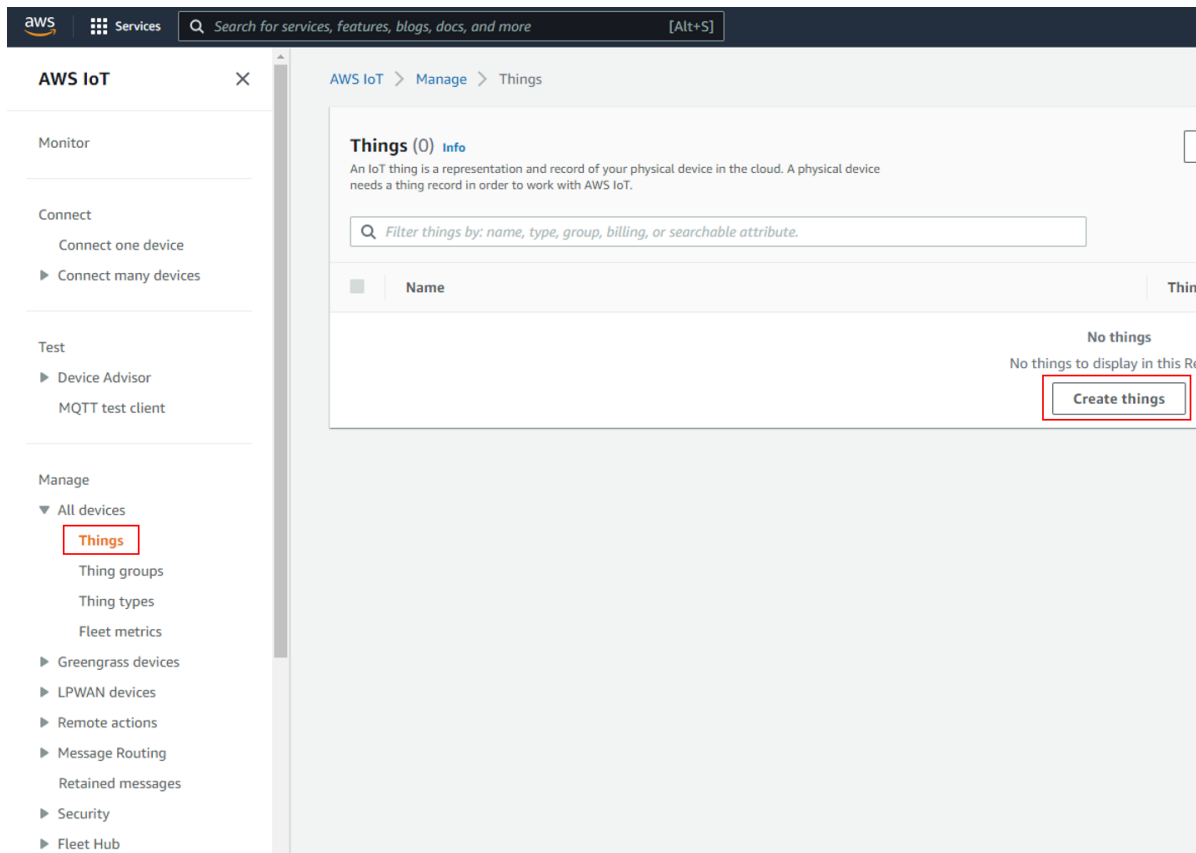


Figure 8: IoT Core: Things



4. Select "Create single thing" and Click "Next".

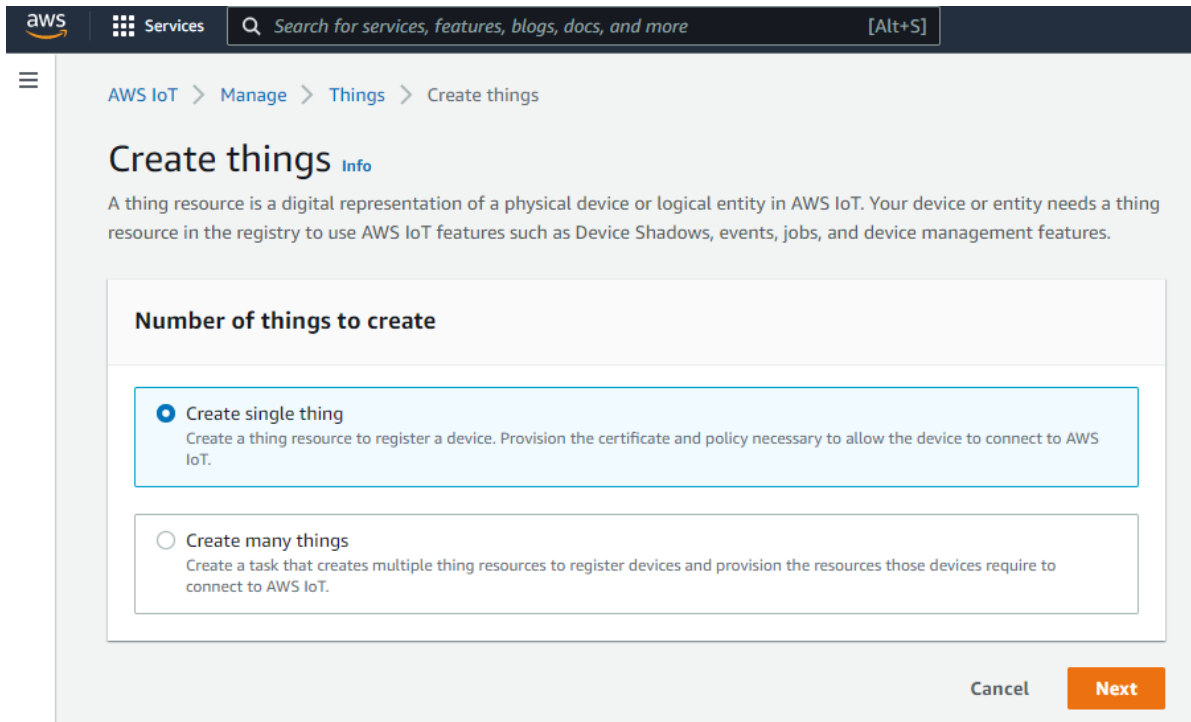


Figure 9: IoT Core: Create single Thing

5. "Specify thing properties". write name of thing in "Thing name".  
For example "AWS\_WE\_ADRASTEIA". Additional configurations are optional, keep them as default.

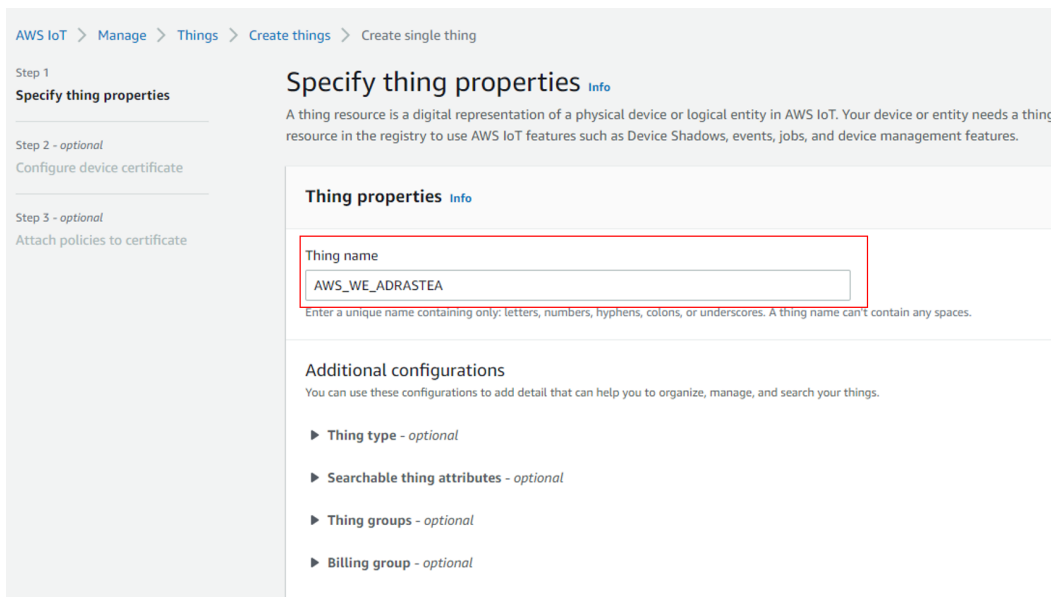
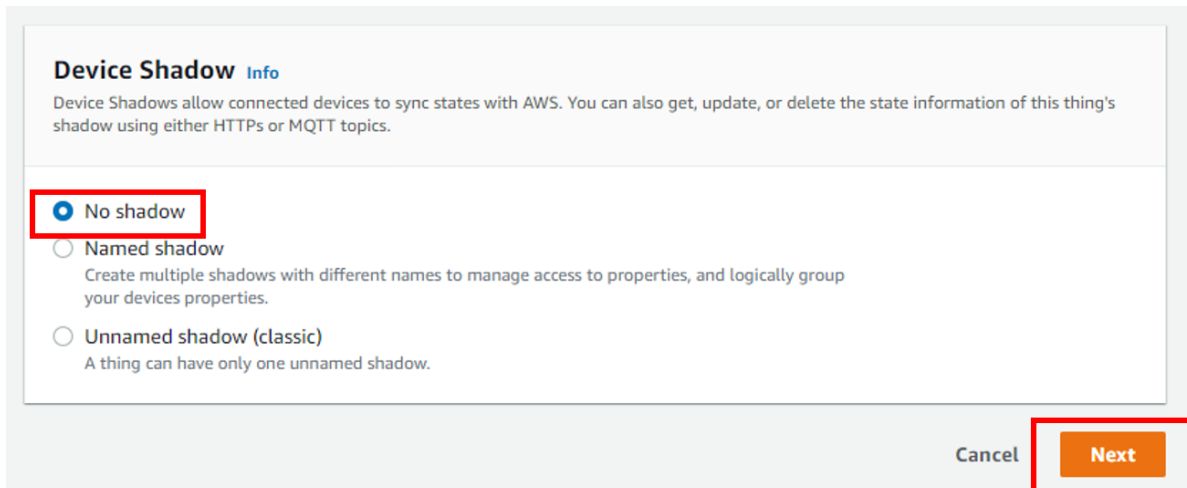


Figure 10: IoT Core: Specify thing properties

6. Keep "Device Shadow" as default to "No shadow". Click "Next".



**Device Shadow** [Info](#)

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPs or MQTT topics.

☒ **No shadow**

☐ **Named shadow**  
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.

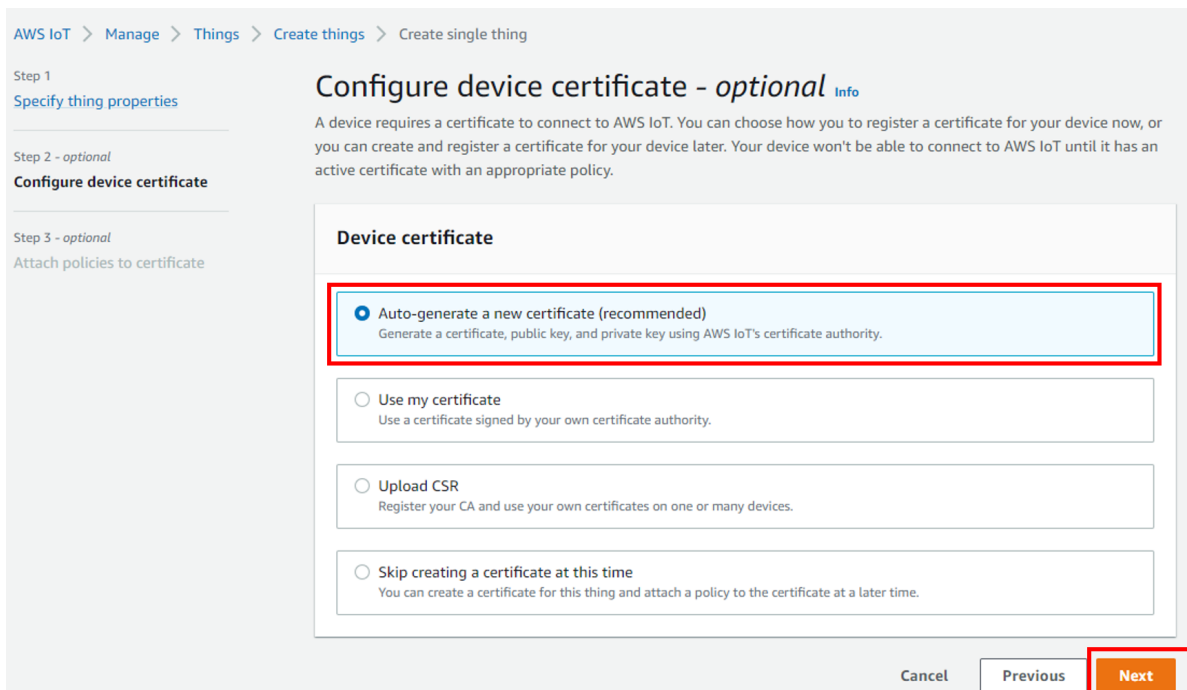
☐ **Unnamed shadow (classic)**  
A thing can have only one unnamed shadow.

Cancel **Next**

Figure 11: IoT Core: Device Shadow

### 3.3 Configuration and Downloading Device Certificate and Key

1. Configure device certificate. Default settings are recommended "Auto-generate a new certificate (recommend)". Click "Next". It will generate a device certificate, a public key and a private key.



AWS IoT > Manage > Things > Create things > Create single thing

Step 1  
[Specify thing properties](#)

Step 2 - optional  
**Configure device certificate**

Step 3 - optional  
[Attach policies to 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.

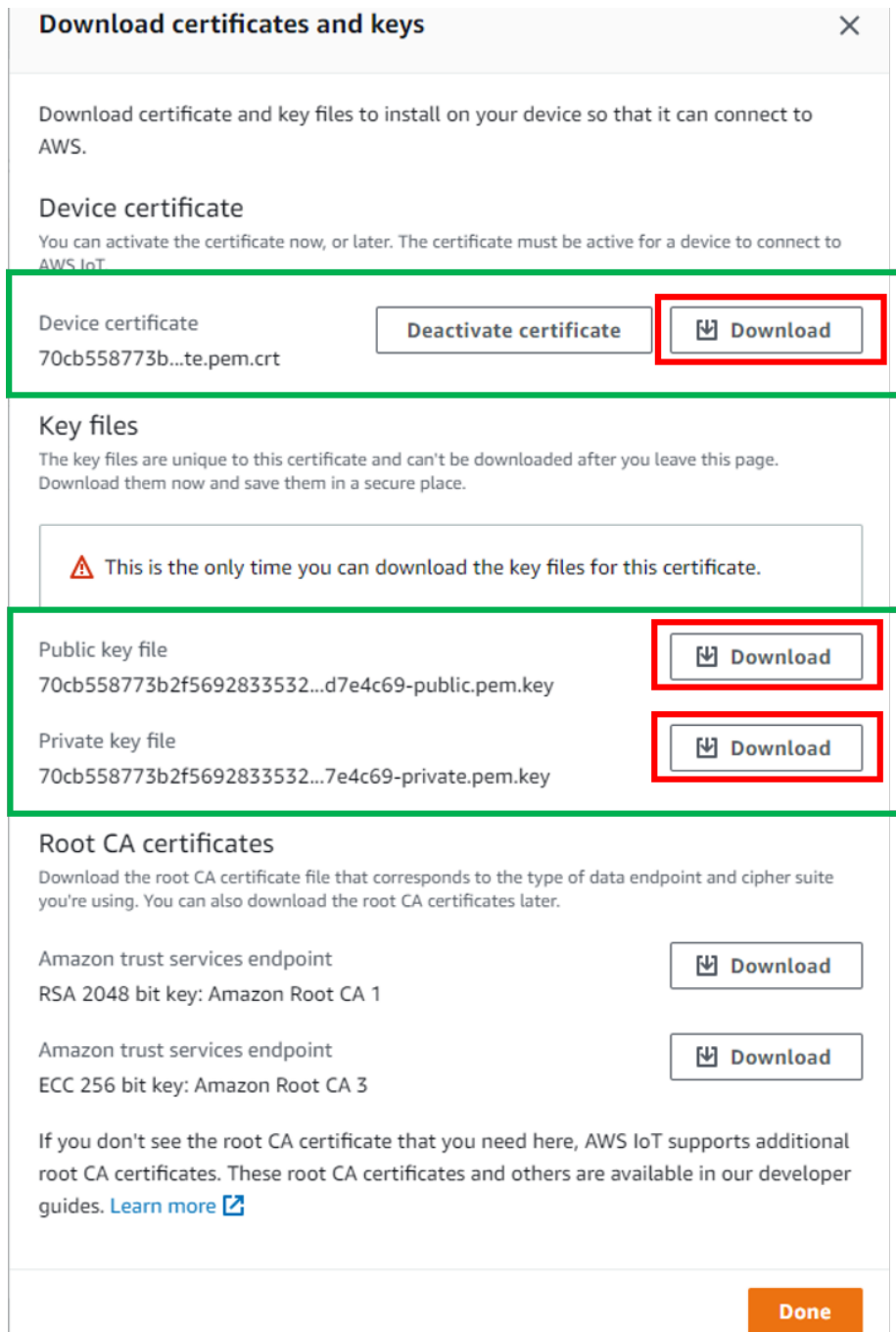
☐ **Upload CSR**  
Register your CA and use your own certificates on one or many devices.

☐ **Skip creating a certificate at this time**  
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel Previous **Next**

Figure 12: IoT Core: Configure Device Certificates

2. Download and keep these files well stored and secured.



**Download certificates and keys**

Download certificate and key files to install on your device so that it can connect to AWS.

**Device certificate**  
You can activate the certificate now, or later. The certificate must be active for a device to connect to AWS IoT.

Device certificate  
70cb558773b...te.pem.crt

Deactivate certificate

Download

**Key files**  
The key files are unique to this certificate and can't be downloaded after you leave this page. Download them now and save them in a secure place.

**This is the only time you can download the key files for this certificate.**

Public key file  
70cb558773b2f5692833532...d7e4c69-public.pem.key

Download

Private key file  
70cb558773b2f5692833532...7e4c69-private.pem.key

Download

**Root CA certificates**  
Download the root CA certificate file that corresponds to the type of data endpoint and cipher suite you're using. You can also download the root CA certificates later.

Amazon trust services endpoint  
RSA 2048 bit key: Amazon Root CA 1

Download

Amazon trust services endpoint  
ECC 256 bit key: Amazon Root CA 3

Download

If you don't see the root CA certificate that you need here, AWS IoT supports additional root CA certificates. These root CA certificates and others are available in our developer guides. [Learn more](#)

Done

Figure 13: IoT Core: Download Certificates and Keys



The root certificate of AWS IoT can found here

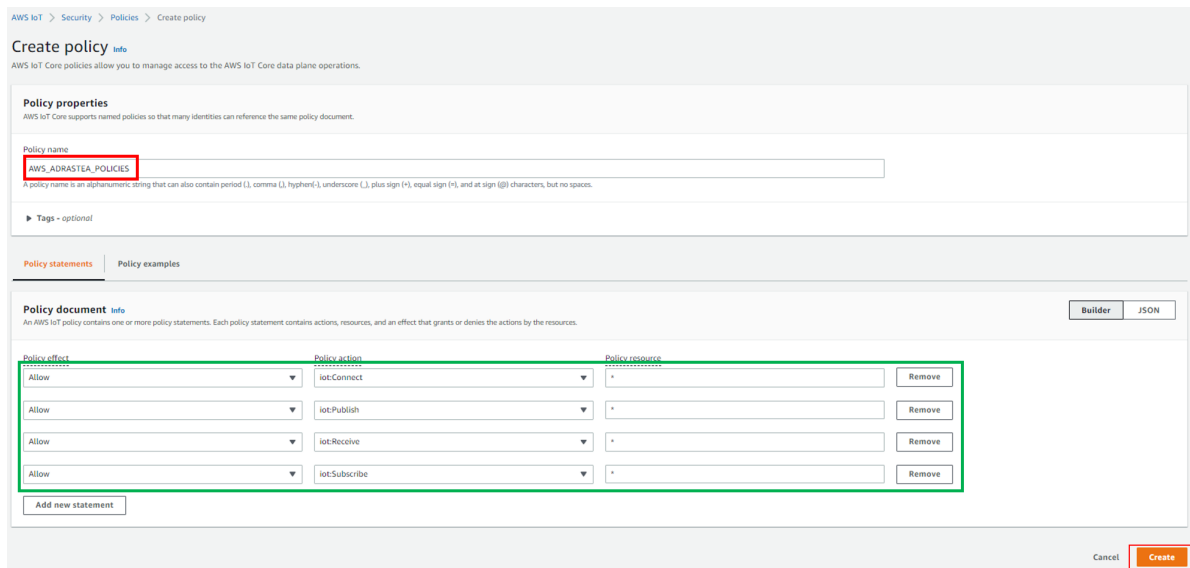
<https://www.amazontrust.com/repository/AmazonRootCA1.pem>

### 3.4 Setting Up and Attaching Policies to Device:

1. Create a policy from Security -> Policies -> Create, fill the name of policy and add statements.



The following block is a recommended statements for reference. Developer can also define his/her own rule.



AWS IoT > Security > Policies > Create policy

#### Create policy Info

AWS IoT Core policies allow you to manage access to the AWS IoT Core data plane operations.

**Policy properties**

AWS IoT Core supports named policies so that many identities can reference the same policy document.

Policy name

**AWS\_ADRASTEIA\_POLICIES**

A policy name is an alphanumeric string that can also contain period (.), comma (,), hyphen (-), underscore (\_), plus sign (+), equal sign (=), and at sign (@) characters, but no spaces.

Tags - optional

**Policy statements** | Policy examples

**Policy document Info**

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.

Builder | JSON

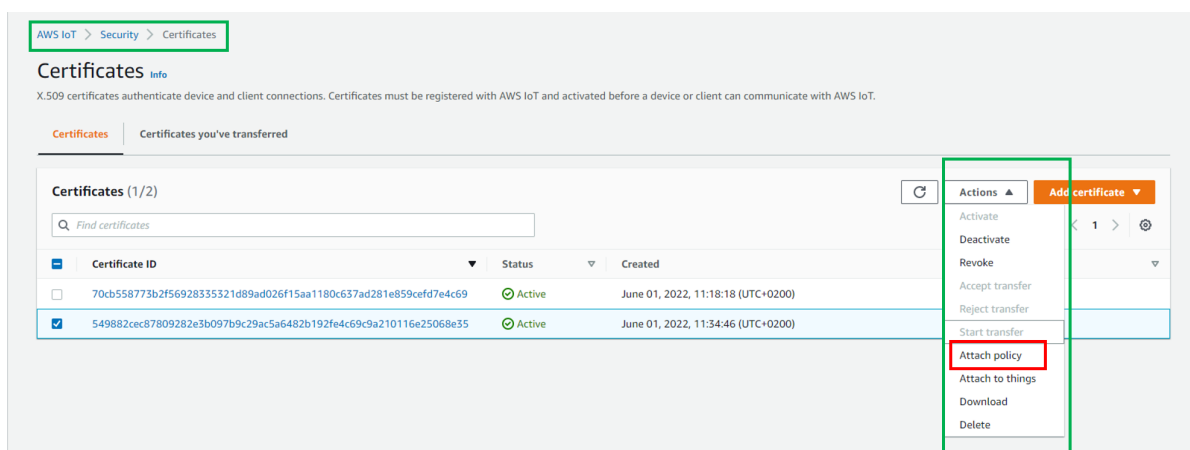
Policy effect	Policy action	Policy resource	
Allow	iot:Connect	*	Remove
Allow	iot:Publish	*	Remove
Allow	iot:Receive	*	Remove
Allow	iot:Subscribe	*	Remove

Add new statement

Cancel **Create**

Figure 14: IoT Core: Create policies

2. Attach policies created in the previous step to the certificates "Security -> Certificates".



AWS IoT > Security > Certificates

#### Certificates Info

X.509 certificates authenticate device and client connections. Certificates must be registered with AWS IoT and activated before a device or client can communicate with AWS IoT.

**Certificates** | Certificates you've transferred

**Certificates (1/2)**

Find certificates

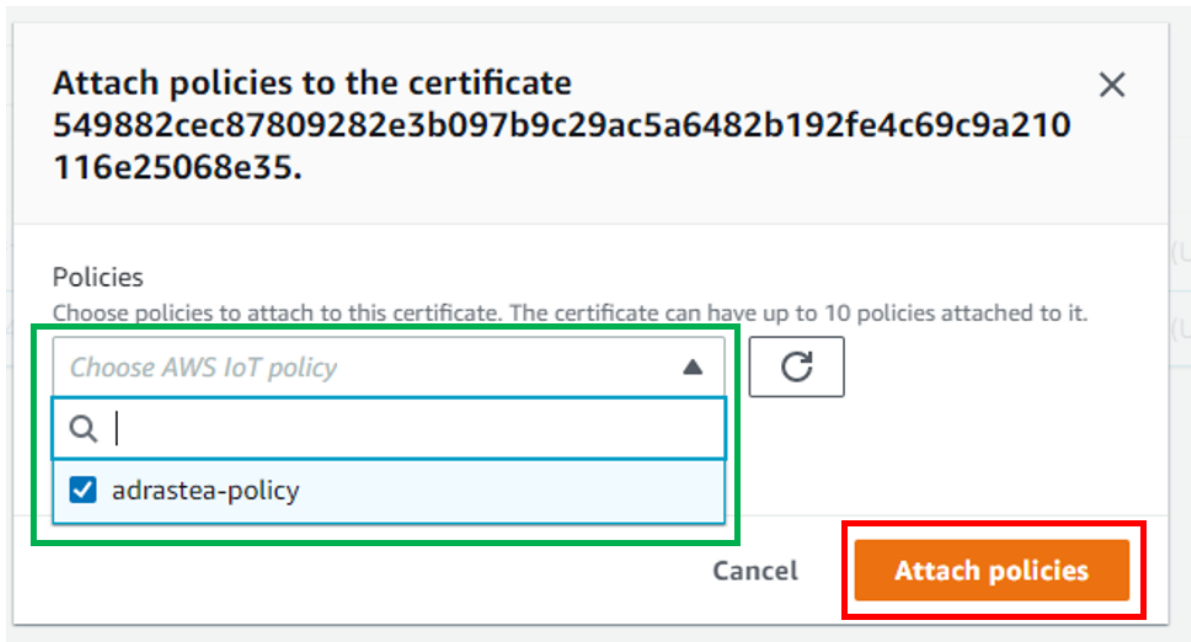
	Certificate ID	Status	Created
<input type="checkbox"/>	70cb558773b2f56928335321d89ad026f15aa1180c637ad281e859cefd7e4c69	Active	June 01, 2022, 11:18:18 (UTC+0200)
<input checked="" type="checkbox"/>	549882cec87809282e3b097b9c29ac5a6482b192fe4c69c9a210116e25068e35	Active	June 01, 2022, 11:34:46 (UTC+0200)

Actions ▴ | Add certificate ▾

- Activate
- Deactivate
- Revoke
- Accept transfer
- Reject transfer
- Start transfer
- Attach policy**
- Attach to things
- Download
- Delete

Figure 15: IoT Core: Select Policy to Attach with Certificate

3. Attach to a pre-defined policies.



**Attach policies to the certificate** ✕

549882cec87809282e3b097b9c29ac5a6482b192fe4c69c9a210  
116e25068e35.

**Policies**  
Choose policies to attach to this certificate. The certificate can have up to 10 policies attached to it.

Choose AWS IoT policy ▲ ↺

Q |

☒ adrastea-policy

Cancel Attach policies

Figure 16: IoT Core: Attach Policy to Certificate



To check the endpoint of things which has been created in above steps follow:  
AWS IoT-> Manage -> Things -> WE\_ADRASTEIA\_AWS

### 3.5 Testing with Device:

1. Click "Test -> MQTT Test Client" on the left bar to test your device.

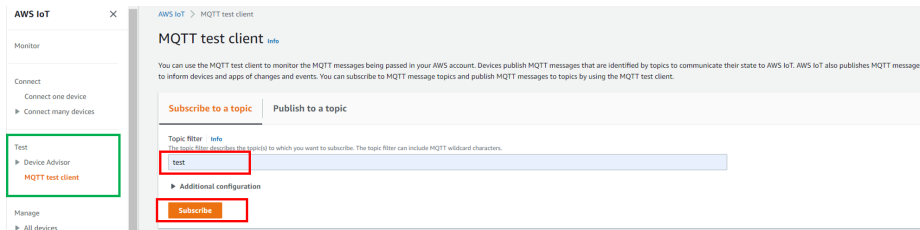


Figure 17: IoT Core: MQTT Test Client

2. Subscribe To Topic "test".

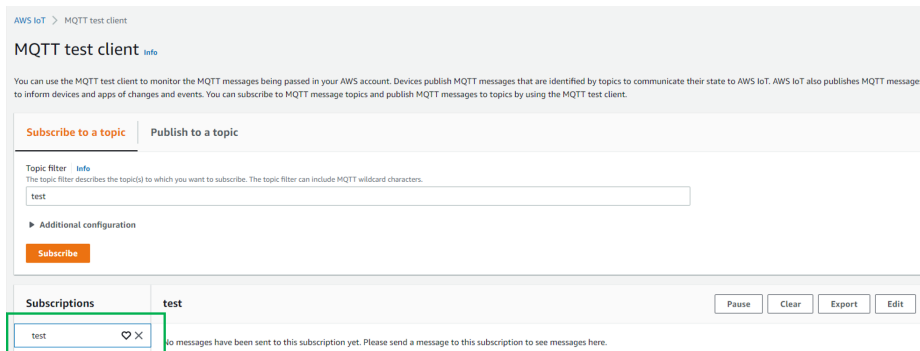


Figure 18: IoT Core: Subscribe To Topic

3. This step is to test downlink data flow from cloud to device to Publish data on topic "test". Follow this steps only when "DATA" is received in uplink from Device to Cloud.

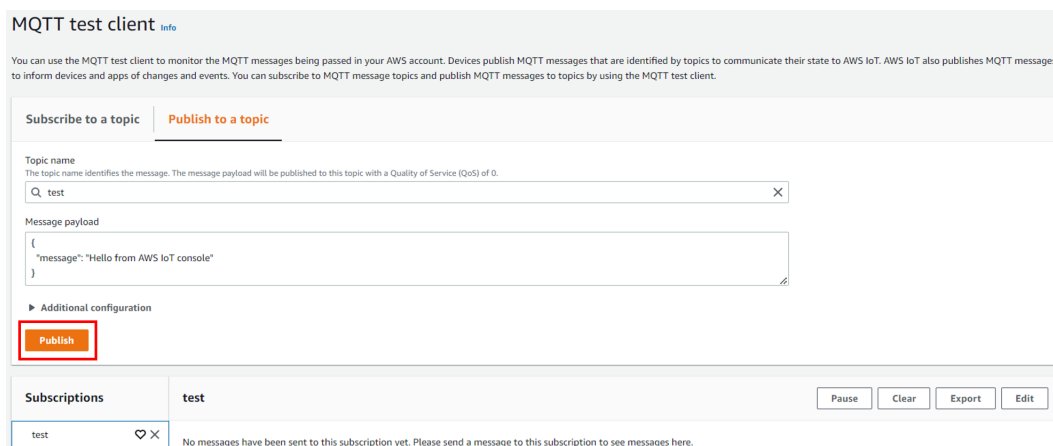


Figure 19: IoT Core: Publish data

## 4 Adrastea To Cloud Examples

The intention here is to go through the process step-by-step to enable the user to replicate the same steps without much efforts. Connection to the most commonly used cloud platforms Amazon AWS is presented. However, the same concept can be extended to work with any cloud platform that supports MQTT.

### 4.1 Adrastea-I to AWS Cloud Example

In this example, the Adrastea-I creates and subscribes a specific topic "test", then sends the data to AWS and receives data from AWS.



Before starting these steps, make sure the steps in Chapter 3 are completed

#### 1. Load the User Certificate:



This is the Device certificate downloaded in Step 2 of 3.3



Certificate and private key shall be in PEM format. Usage of quotes is mandatory.



Type and enter "map" command, this command is required before sending the AT commands to modem.



When AT Comamnd UART port is opened Adrastea-I output console displays:  
%SCMNOTIFYEV:"ADRASTEAIReady"

```
at%certcmd="write","ca—certificate.pem.crt",0,"-----BEGIN_CERTIFICATE
-----*****-----END_CERTIFICATE-----"
```

OK

#### 2. Load the user private key:



This is the Private key file downloaded in Step 2 of 3.3

```
at%certcmd="write","ca-private.pem.key",1,"-----BEGIN_RSA_PRIVATE_KEY
-----*****-----END_RSA_PRIVATE_KEY-----"
```

OK



The data content in PEM format is transferred in pseudo-text format with <LF> (0x10) service symbols inside and will be shown with newlines. When using the AT%CERTCMD="WRITE" command, avoid CR characters while pasting command. To do so, copy the command to an editor that supports EOL (end-of-line) format conversion and convert it to UNIX format (i.e., LF format). Then copy and paste this converted content to Adrastea Commander/PuTTY/teraterm.

3. Read the Device Certificate, this is optional step to verify contents of device certificate:

```
AT%CERTCMD="READ","ca-certificate.pem.crt"
%CERTCMD:"-----BEGIN_CERTIFICATE-----*****-----END
_CERTIFICATE-----"
```

OK

4. Add certificates to the profile:

```
AT%CERTCFG="ADD",1,,,"ca-certificate.pem.crt","ca-private.pem.key"
OK
```

5. Allow AWS events:

```
AT%AWSIOTEV="ALL",1
OK
```

6. Connect an AWS session:



To check the endpoint of things which has been created in above steps follow:  
AWS IoT-> Settings-> Device data endpoint

```
AT%AWSIOTCFG="CONN","a18jcdjlx073x-ats.iot.eu-central-1.amazonaws.com"
,1,"WE_ADRASTEIA_AWS"
OK
```



7. Configure AWS IOT cloud connection parameters:

```
AT%AWSIOTCFG="PROTOCOL",1200,1  
OK
```

8. Connect to the server:

```
AT%AWSIOTCMD="CONNECT"  
OK
```

9. Wait to get the connection event:

```
%AWSIOTEVU:"CONCONF",0
```

10. Subscribe to topic "test":

```
AT%AWSIOTCMD="SUBSCRIBE","test"  
%AWSCMD: 1  
OK
```

11. Get subscription success notification:

```
%AWSIOTEVU:"SUBCONF",1,0
```

12. Publish some "DATA" to topic "test":

```
AT%AWSIOTCMD="PUBLISH","test","DATA"  
%AWSCMD: 2  
OK
```

13. Get the Notification:

```
%AWSIOTEVU:"PUBCONF",1,0
```

14. Received data from the Server:

```
%AWSIOTEVU:"PUBRCV","test","{\"message\": \"Hello From AWS\"}"
```

## 5 MQTT Related AT Commands

### 5.1 MQTT AT Commands

#### 5.1.1 %MQTTCFG: Configure MQTT Connection Parameters

AT command to configure MQTT connection parameters.

Command	Command Type	Response
AT%MQTTCFG=<obj>,<conn_id>[, [<param1>][,<param2>]...]	Set	OK ERROR
AT%MQTTCFG?	Read	ERROR
AT%MQTTCFG=?	Test	%MQTTCFG: (list of supported <cmd>),(list of supported <conn_id>)

Table 1: AT%MQTTCFG

#### Description:

AT command to configure MQTT connection parameters. To start new MQTT connection the "NODES" parameters shall be defined at least. Other configurations may be omitted, default settings are use :

- If "TLS" layer is not configured, unsecured connection will be established by default.
- If "IP" layer is not configured, default PDN, IP type and default MQTT ports will be used.
- If "PROTOCOL" parameters are not configured, default protocol parameters will be selected.
- If "WILLMSG" parameters are not configured, no Will message will be used.

To make this omission confidentially working, it is strictly recommended to call "CLEAR" sub-command before entering new configuration for previously used or default <conn\_id>.

Connection ID parameter is introduced to handle multi-connection MQTT. Use zero value for <conn\_id> if single connection is expected. The ID for multi-connection is assigned by user and then used for all connection configuration in current AT%MQTTCFG, command (AT%MQTTCMD) and event (AT%MQTTEV/%MQTTEVU).

**Defined values:****<obj>**: string.

Value	Description
"NODES"	Configure client & server nodes parameters
"TLS"	Configure TLS layer security parameters
"IP"	Configure IP layer parameters
"WILLMSG"	Configure MQTT will message
"PROTOCOL"	Configure MQTT protocol parameters
"CLEAR"	Clear all previous settings for specified <conn_id>

Table 2: MQTTCMD\_Obj\_Description

**<conn\_id>**: integer. Default or previously assigned <conn\_id>.

- 0: Single MQTT connectivity mode.
- 1 -5: Multi-connected mode.

**For "NODES" :****<param1>**: string. Unique client ID used to connect to the broker.**<param2>**: string. Broker URL or IP address.**<param3>**: string. Optional username for broker authentication.**<param4>**: string. Optional password for broker authentication.**For "TLS" :****<param1>**: string. TLS authentication mode;

- 0: mutual authentication (default).
- 1: Authenticate client side only.
- 2: Authenticate server side only.

**<param2>**: integer. TLS predefined authentication context (profile) previously configured by AT%CERTCFG. Default zero profile ID may be used for server authentication only and will apply root CAs stored into Root Trusted folder for authentication.

**For "IP" :**

**<param1>**: integer. Optional Session ID. Numeric PDN identification defined in APN table for specified PDN. If Session ID=0 or omitted default data PDN is used unless configured differently by AT%SETRROUTE;

- 0: Use default data PDN.
- 1: max value defined in NP config file.

**<param2>**: integer. Optional IP type used to configure preferred IP type for connection.

- 0: IPv4v6 (default).
- 1: IPv4.
- 1: IPv6.

**<param3>**: integer. Optional destination (server) TCP/UDP port number. If omitted default MQTT port number is used. Value range 1 - 65535.

**For "WILLMSG":**

**<param1>**: integer. Will message presence.

- 0: disable (default value).
- 1: enable.

**<param2>**: integer. Will QoS value.

- 0: at most once delivery (default value).
- 1: at least once delivery.
- 1: exactly once delivery.

**<param3>**: integer. Will message retain - whether or not the Will Message will be retained across disconnects.

- 0: (default value): the Will Message will not be retained at the MQTT server across disconnects from MQTT client
- 1: the Will Message will be retained by the MQTT server across disconnects from MQTT client (until superseded by another message).

**<param4>**: string. Will Topic - Standard MQTT Topic Name. It could include various Topic Separators "/" to form various Topic levels.

**<param5>**: string. The Will message defines the content of the message that is published to the Will topic if the client is unexpectedly disconnected.

**For "PROTOCOL":**

**<param1>**: integer. MQTT protocol type for connection.

- 0:MQTT (default).

**<param2>**: integer. Keep-alive time. The default value is 600 sec (10 min). Unit: second. It defines the maximum time interval between messages received from a client.

- 0: no timeout, keep-alive deactivated.
- 1 - 65535 (18 hours, 12 minutes and 15 seconds.).

**<param3>**: integer. Clean session type.

- 0: the server must store the subscriptions of the client after it disconnects.
- 1: the server must discard any previously maintained information about the client and treat the connection as "clean". Default policy.

### 5.1.2 %MQTTCMD: Communicate With MQTT Server (Broker)

Command	Command Type	Response
AT%MQTTCMD=<cmd>, <conn_id>[,<param1>, <param2>[,<param3>[, <param4>][,<param5>]]] [<data>]	Set	%MQTTCMD:<"CONNECT"/"DISCONNECT"/ "SUBSCRIBE"/"UNSUBSCRIBE"/"PUBLISH" /"PUBACK"/"REGISTER">,<conn_id>[, [<param1>][,<param2>]...]  OK
AT%MQTTCMD?	Read	ERROR
AT%MQTTCMD=?	Test	%MQTTCMD: (list of supported <cmd>),(list of supported <conn_id>)

Table 3: AT%MQTTCMD

#### Description:

AT command to communicate with MQTT server (broker). All commands are un-blocking. The information about command success or fail will be provided in %MQTTEVU URC. The Will message used in "CONNECT" shall be predefined in AT%MQTTCFG.

The "PUBRCV" URC can provide incoming publication data in the <data> parameter only for textual or pseudo-textual data transfer (i.e. JSON, PEM, B64, etc.). The arbitrary binary data transfer is possible only to file. Use AT%MQTTCMD="SUBSCRIBE" to define filename for binary data download.

The "PUBLISH" command provides 2 mechanisms to publish data.

- Only textual or pseudo-textual (i.e. JSON, PEM, B64, etc.) data transfer is permitted for direct AT call using <data> parameter.
- The arbitrary binary data transfer is possible only from file.

For non-file "PUBLISH" operation the data size parameter <param4> may be omitted in human debug mode of AT usage. In this use-case Ctrl+Z pressing shall signal data end.

The "SUBSCRIBE" with defined filename parameter will cause that all following server publications will be stored into the file and signaled by %MQTTEVU: "PUBRCV" URC. Use different filenames for different <conn\_id> and topic names to prevent file override, if needed. The file for server publication will be always located on temporary storage disk b:/. User shall specify only filename for "SUBSCRIBE" sub-command. Any attempt to specify full path in this command will be rejected with ERROR.

#### Defined values:

<cmd>: string.

Value	Description
"CONNECT"	Start connection with endpoint
"DISCONNECT"	End connection with endpoint
"SUBSCRIBE"	Subscribe to a topic on the endpoint
"UNSUBSCRIBE"	Stop subscription to a topic on the endpoint.
"PUBLISH"	Send publish packet to endpoint
"PUBACK"	Not support
"REGISTER"	Not support

Table 4: %MQTTCMD\_Cmd\_Description

**<conn\_id>**: integer. Default or previously assigned <conn\_id>.

- 0: single MQTT connectivity mode. Each configuration overrides previous setting
- 1 - 5: multi-connected mode.

**<msg\_id>**: message ID. Value range 1-65535.

#### For "CONNECT"/"DISCONNECT"

No <param>

#### For "SUBSCRIBE" :

**<param1>**: integer. The QoS level at which the client wants to publish the message.

- 0: At most once delivery (default value).
- 1: At least once delivery.
- 2: Exactly once delivery.

**<param2>**: string. The subscription topic name.

**<param3>**: string. Optional parameter. Filename to store received publications on b:/.

#### For "UNSUBSCRIBE" :

**<param1>**: string. The subscription topic name.

#### For "PUBLISH":

**<param1>**: integer. The QoS level at which the client wants to publish the message.

- 0: At most once delivery (default value).
- 1: At least once delivery.
- 2: Exactly once delivery.

**<param2>**: integer. Whether or not the server will retain the message after it has been delivered to the current subscribers.

- 0: The server will not retain the message after it has been delivered to the current subscribers.
- 1: The server will retain the message after it has been delivered to the current subscribers.

**<param3>**: string. The publication topic name.

**<param4>**: integer. Actual data size in bytes for transfer to server.

- 0: Undefined, publish from file.
- 1: 3000

**<param5>**: string. Optional parameter. Full path to file to publish from.

**<data>**: string MQTT raw data payload without quotes.

### 5.1.3 %MQTTEV: Notify About MQTT Events

Command	Command Type	Response
AT%MQTTEV= <ev_type>,<mode>	Set	OK or ERROR
AT%MQTTEV?	Read	ERROR
AT%MQTTEV=?	Test	%MQTTEV: (list of supported <ev_type>), (list of supported <mode>)
Unsolicited	Unsolicited	%MQTTEVU:<ev_type>,<conn_id>,<res1>[,<res2>[,<res3>[,<res4>,<res5>]]][<data>]

Table 5: AT%MQTTEV

#### Description:

The command is intended to notify about MQTT events. Default MQTT mode is URC disabled for all event types except of "PUBRCV", which is enabled by first call of AT%MQTTCMD="SUBSCRIBE". Most of the events are related to asynchronous operation triggered by AT%MQTTCMD. Such acknowledgement may be normally disabled.

Only "PUBRCV" event provides the data from the topic, to which the client was pre-subscribed by AT%MQTTCMD="SUBSCRIBE".

The "PUBRCV" URC can provide incoming publication data in the <data> parameter only for textual or pseudo-textual (i.e. JSON, PEM, B64, etc.) data transfer. The arbitrary binary data transfer is possible only to file on b:/. Use AT%MQTTCMD="SUBSCRIBE" to define filename for binary data download.



Note: AT%MQTTCMD="PUBLISH" with QoS=0 will not send any acknowledge message and <ev\_type>="PUBCONF" is not expected.

#### Defined values:

<ev\_type>: string.



Value	Description
"CONCONF"	Connect procedure confirmation status
"DISCONF"	Graceful disconnect procedure confirmation status
"SUBCONF"	Subscribe procedure confirmation status
"UNSCONF"	Unsubscribe procedure confirmation status
"PUBCONF"	Outgoing publication procedure confirmation status. Optional URC, depends on "PUBLISH" QoS selected
"PUBRCV"	Incoming publication message received
"CONNFAIL"	Connection failure
"PUBRCVSTORFAIL"	Storage failure of received publication. Ordinary if disk out of space or file is opened for writing.
"ALL"	All events, used only in execution command

Table 6: %MQTTCMD\_Ev\_Type\_Description

**<mode>**: integer. Status of unsolicited result response presentation.

- 0: disable (default value).
- 1: enable.

**<conn\_id>**: integer. Default or previously assigned <conn\_id>.

- 0: Single MQTT connectivity mode.
- 1 -5: Connection ID in multi-connected mode

**For "CONCONF"/"DISCONF" :**

**<res1>**: integer. Result code.

- 0: Success
- 1: Fail

**<res2>**: integer. Optional error code.

- 0: No Error
- 1: Error

**For "UNSCONF" :**

**<res1>**: integer. Message ID value range 1-65535.

**<res2>**: integer. Result code.

- 0: Success
- 1: Fail

**<res3>**: integer. Optional error code.

**For "SUBCONF"/"PUBCONF" :**

**<res1>**: Message ID value range 1-65535.

**<res2>**: integer. Result code.

- 0: Success
- 1: Fail

**<res3>**: integer. Optional error code.

**For "PUBRCV"/"PUBRCVSTORFAIL" :**

**<res1>**: Message ID. It may be zero (undefined) for QoS=0 ;

- 0: Undefined
- 1-65535

**<res2>**: string. The publication topic name.

**<res3>**: integer. Data size in bytes transferred by this URC. If this parameter is equal to zero, no any **<data>** arrival is expected in the same URC.

**<res4>**: integer. Optional data size in bytes stored into file.

**<res5>**: string. Optional parameter. Filename, where received publication has been stored (or attempted to be stored for "PUBRCVSTORFAIL") on b:/.

**<data>**: string. MQTT raw data payload without quotes.

## 5.2 MQTT AT Commands for AWS

### 5.2.1 %AWSIOTCFG: Configure AWS IoT Cloud

AT command to configure AWS IOT cloud connection parameters.

Command	Command Type	Response
AT%AWSIOTCFG=<cmd>, [<param1>][,<param2> [,<param3>]]	Set	OK ERROR
AT%AWSIOTCFG?	Read	ERROR
AT%AWSIOTCFG=?	Test	%AWSIOTCFG: (list of supported <cmd>)

Table 7: AT%AWSIOTCFG

#### Description:

AT command to configure AWS IOT cloud connection parameters. To start new AWS IOT connection the "CONN" parameters shall be defined at least. Mandatory TLS profile ID, which shall be pre-configured by AT%CERTCFG, is a special TLS profile, which does not contain both.: root certificate file and root certificate path. The root certificate path is hardcoded in SW and implies the usage of trusted root CA pre-installed into device to support proper AWS security level. If selected TLS certificate profile contains <ca\_file> or <ca\_path> fields (see AT%CERTCFG), AT command returns ERROR. If "PROTOCOL" parameters are not configured, default protocol parameters will be selected.

#### Defined values:

<cmd>: string.

Value	Description
"CONN"	Pre-configure connection parameters
"PROTOCOL"	Pre-configure protocol parameters
"IP"	IP Layer parameters.

Table 8: %AWSIOTCFG\_Cmd\_Description

**For "CONN":** Pre-configure connection parameters

<param1>: string. Endpoint URL.

<param2>: integer. TLS predefined authentication context (profile) previously configured by AT%CERTCFG.

<param3>: string. Optional unique client ID used to connect to the broker. The IMEI is used as client ID by default.

**For "PROTOCOL":**Pre-configure protocol parameters

**<param1>:** integer. Optional MQTT keep-alive time in seconds. Default 1200 (20 min). Value range 1-1200.

**<param2>:** integer. Optional QoS setting for "PUBLISH".

- 0: With no confirmation (default value)
- 1: Confirmed (acknowledged)

**For "IP":** IP layer parameters.

**<param1>:** integer. Optional Session ID: numeric PDN identification defined in APN table for specified PDN. If Session ID=0 or omitted default data PDN is used unless configured differently by AT%SETROUTE:

- 0: use default data PDN
- 1: max value defined in NP config file

**<param2>:** Integer. Optional IP type used to configure preferred IP type for connection.

- 0: IPv4v6 (default).
- 1: IPv4.
- 1: IPv6.

### 5.2.2 %AWSIOTCMD: Communicate with AWS IoT Message Broker

AT command to communicate with AWS IoT message broker.

Command	Command Type	Response
AT%AWSIOTCMD=<cmd> [,<param1> [,<param2>]]	Set	For "SUBSCRIBE"/"UNSUBSCRIBE"/"PUBLISH": [%AWSIOTCMD: <msg_id>]
AT%AWSIOTCMD?	Read	ERROR
AT%AWSIOTCMD=?	Test	%AWSIOTCMD: (list of supported <cmd>s)

Table 9: AT%AWSIOTCMD

#### Description:

AT command to communicate with AWS IoT message broker.

All commands are unblocking.

The information about command success or fail will be provided in %AWSIOTEVU URC.

Non-zero message ID may be used to pair subscribe, unsubscribe and publish (confirmed) messages with their URCs. At this stage, message ID is not supported, zero value is returned.

#### Defined values:

<cmd>: string.

Value	Description
"CONNECT"	Start connection with endpoint
"DISCONNECT"	End connection with endpoint
"SUBSCRIBE"	Subscribe to a topic on the endpoint
"UNSUBSCRIBE"	Stop subscription to a topic on the endpoint.
"PUBLISH"	Send publish packet to endpoint

Table 10: %AWSIOTCMD\_Cmd\_Description

**For "SUBSCRIBE":** Subscribe (register) to the topic on the endpoint.

<param1>: string. The subscription topic name.

**For "UNSUBSCRIBE":** Stop subscription (unregister) from the topic on the endpoint

<param1>: string. The subscription topic name.

<cmd> **For "PUBLISH":** Publish (send) packet to endpoint

<param1>: string. The publication topic name.

<param2>: string. Message that appears in the publication, max length is supported 3000(1500bytes).

<msg\_id>: message ID.

- 0: Not in use

- 1-65535

### 5.2.3 %AWSIOTEV: Notify About AWS IOT Events

The command is intended to notify about AWS IOT events.

Command	Command Type	Response
AT%AWSIOTEV==<ev_type><mode>	Set	OK ERROR
AT%AWSIOTEV?	Read	ERROR
AT%AWSIOTEV=?	Test	%AWSIOTEV: (list of supported <ev_type>s), (list of supported <mode>s)
Unsolicited	Unsolicited	%AWSIOTEVU:<ev_type>,<res1>[,<res2> [,<res3]]

Table 11: AT%AWSIOTEV

#### Description:

The command is intended to notify about AWS IOT events. Default mode is URC disabled for all event types except of "PUBRCV", which is enabled by first call of AT%AWSIOTCMD="SUBSCRIBE". Most of the events are related to asynchronous operation triggered by AT%AWSIOTCMD. Such acknowledgement may be normally disabled.

Only "PUBRCV" event provides the data from the topic, to which the client was pre-subscribed (pre-registered) by AT%AWSIOTCMD="SUBSCRIBE".



Note: AT%AWSIOTCMD="PUBLISH" in unconfirmed mode (no ACK) will not send any acknowledge message and <ev\_type>="PUBCONF" is not expected.

Non-zero message ID may be used to pair subscribe, unsubscribe and publish (confirmed) messages sent by AT%AWSIOTCMD with their URCs. At this stage, message ID is not supported, zero value is always reported.



Note: If TCP session is disconnected because of link lost, no URC is sent.

#### Defined values:

<ev\_type>: string.

Value	Description
"CONCONF"	Connect procedure confirmation status
"DISCONF"	Graceful disconnect procedure confirmation status
"SUBCONF"	Subscribe procedure confirmation status
"UNSCONF"	Unsubscribe procedure confirmation status
"PUBCONF"	Outgoing publication procedure confirmation status. Optional URC, depends on "PUBLISH" QoS selected
"PUBRCV"	Incoming publication message received
"CONNFAIL"	Connection failure
"PUBRCVSTORFAIL"	Storage failure of received publication. Ordinary if disk out of space or file is opened for writing.
"ALL"	All events, used only in execution command

Table 12: %AWSIOTEV\_Ev\_Type\_Description

**<mode>**: integer. Status of unsolicited result response presentation.

- 0: disable (default value).
- 1: enable.

**For "CONCONF"/"DISCONF" :**

**<res1>**: integer. result code.

- 0: Success
- 1: Fail

**For "SUBCONF"/"UNSCONF"/"PUBCONF":**

**<res1>**: Message ID.

- 0: not in use
- 1-65535

**<res2>**: integer. Result code.

- 0: Success
- 1: Fail

**For "PUBRCV":**

**<res1>**: string. The publication topic name.

**<res2>**: string. Publication message content received from endpoint.



## 6 Summary

Designing an IoT solution brings with it a number of challenges. Being multifaceted, IoT applications demands a lot of competence from hardware design to UI development. Under such situations, it is prudent to take a modular approach. This means breaking down the architecture into functionally independent blocks. One such essential building-block for an IoT application is wireless connectivity. Integrating a radio module such as Adrastea-I enables the designer to completely delegate the task of wireless connectivity, which saves a lot of effort enabling quicker time-to-market.

At the other end, it is a completely different approach necessary to arrive at a cloud solution. Flexibility and security are the key parameters that needs to be kept in mind while designing a cloud solution.

In this application note, a simple IoT technology stack that represents the principle behind any IoT solution is presented. Further, with the help of an example, a solution using the Adrastea-I Cellular module and AWS cloud platform is described. Finally, a step-by-step prototype of the proposed solution is presented. Scaling this manual process to a huge quantity of products is a challenge. This problem can be solved by writing firmware in ARM Cortex M4 available in Adrastea-I cellular module for application development.

## 7 References

- [1] MQTT Amazon AWS. <https://docs.aws.amazon.com/iot/latest/developerguide/mqtt.html>.
- [2] MQTT Microsoft Azure. <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support>.
- [3] MQTT specification (V3.1.1). <https://mqtt.org/mqtt-specification/>.

## **8 Important notes**

The following conditions apply to all goods within the wireless connectivity product range of Würth Elektronik eiSos GmbH & Co. KG:

### **8.1 General customer responsibility**

Some goods within the product range of Würth Elektronik eiSos GmbH & Co. KG contain statements regarding general suitability for certain application areas. These statements about suitability are based on our knowledge and experience of typical requirements concerning the areas, serve as general guidance and cannot be estimated as binding statements about the suitability for a customer application. The responsibility for the applicability and use in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to the customer to evaluate, where appropriate to investigate and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for the respective customer application or not. Accordingly, the customer is cautioned to verify that the documentation is current before placing orders.

### **8.2 Customer responsibility related to specific, in particular safety-relevant applications**

It has to be clearly pointed out that the possibility of a malfunction of electronic components or failure before the end of the usual lifetime cannot be completely eliminated in the current state of the art, even if the products are operated within the range of the specifications. The same statement is valid for all software sourcecode and firmware parts contained in or used with or for products in the wireless connectivity and sensor product range of Würth Elektronik eiSos GmbH & Co. KG. In certain customer applications requiring a high level of safety and especially in customer applications in which the malfunction or failure of an electronic component could endanger human life or health, it must be ensured by most advanced technological aid of suitable design of the customer application that no injury or damage is caused to third parties in the event of malfunction or failure of an electronic component.

### **8.3 Best care and attention**

Any product-specific data sheets, manuals, application notes, PCN's, warnings and cautions must be strictly observed in the most recent versions and matching to the products firmware revisions. This documents can be downloaded from the product specific sections on the wireless connectivity homepage.

### **8.4 Customer support for product specifications**

Some products within the product range may contain substances, which are subject to restrictions in certain jurisdictions in order to serve specific technical requirements. Necessary information is available on request. In this case, the field sales engineer or the internal sales person in charge should be contacted who will be happy to support in this matter.

## **8.5 Product improvements**

Due to constant product improvement, product specifications may change from time to time. As a standard reporting procedure of the Product Change Notification (PCN) according to the JEDEC-Standard, we inform about major changes. In case of further queries regarding the PCN, the field sales engineer, the internal sales person or the technical support team in charge should be contacted. The basic responsibility of the customer as per section 8.1 and 8.2 remains unaffected. All wireless connectivity module driver software "wireless connectivity SDK" and its source codes as well as all PC software tools are not subject to the Product Change Notification information process.

## **8.6 Product life cycle**

Due to technical progress and economical evaluation we also reserve the right to discontinue production and delivery of products. As a standard reporting procedure of the Product Termination Notification (PTN) according to the JEDEC-Standard we will inform at an early stage about inevitable product discontinuance. According to this, we cannot ensure that all products within our product range will always be available. Therefore, it needs to be verified with the field sales engineer or the internal sales person in charge about the current product availability expectancy before or when the product for application design-in disposal is considered. The approach named above does not apply in the case of individual agreements deviating from the foregoing for customer-specific products.

## **8.7 Property rights**

All the rights for contractual products produced by Würth Elektronik eiSos GmbH & Co. KG on the basis of ideas, development contracts as well as models or templates that are subject to copyright, patent or commercial protection supplied to the customer will remain with Würth Elektronik eiSos GmbH & Co. KG. Würth Elektronik eiSos GmbH & Co. KG does not warrant or represent that any license, either expressed or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, application, or process in which Würth Elektronik eiSos GmbH & Co. KG components or services are used.

## **8.8 General terms and conditions**

Unless otherwise agreed in individual contracts, all orders are subject to the current version of the "General Terms and Conditions of Würth Elektronik eiSos Group", last version available at [www.we-online.com](http://www.we-online.com).

## 9 Legal notice

### 9.1 Exclusion of liability

Würth Elektronik eiSos GmbH & Co. KG considers the information in this document to be correct at the time of publication. However, Würth Elektronik eiSos GmbH & Co. KG reserves the right to modify the information such as technical specifications or functions of its products or discontinue the production of these products or the support of one of these products without any written announcement or notification to customers. The customer must make sure that the information used corresponds to the latest published information. Würth Elektronik eiSos GmbH & Co. KG does not assume any liability for the use of its products. Würth Elektronik eiSos GmbH & Co. KG does not grant licenses for its patent rights or for any other of its intellectual property rights or third-party rights.

Notwithstanding anything above, Würth Elektronik eiSos GmbH & Co. KG makes no representations and/or warranties of any kind for the provided information related to their accuracy, correctness, completeness, usage of the products and/or usability for customer applications. Information published by Würth Elektronik eiSos GmbH & Co. KG regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof.

### 9.2 Suitability in customer applications

The customer bears the responsibility for compliance of systems or units, in which Würth Elektronik eiSos GmbH & Co. KG products are integrated, with applicable legal regulations. Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of Würth Elektronik eiSos GmbH & Co. KG components in its applications, notwithstanding any applications-related information or support that may be provided by Würth Elektronik eiSos GmbH & Co. KG. Customer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences lessen the likelihood of failures that might cause harm and take appropriate remedial actions. The customer will fully indemnify Würth Elektronik eiSos GmbH & Co. KG and its representatives against any damages arising out of the use of any Würth Elektronik eiSos GmbH & Co. KG components in safety-critical applications.

### 9.3 Trademarks

AMBER wireless is a registered trademark of Würth Elektronik eiSos GmbH & Co. KG. All other trademarks, registered trademarks, and product names are the exclusive property of the respective owners.

### 9.4 Usage restriction

Würth Elektronik eiSos GmbH & Co. KG products have been designed and developed for usage in general electronic equipment only. This product is not authorized for use in equipment

where a higher safety standard and reliability standard is especially required or where a failure of the product is reasonably expected to cause severe personal injury or death, unless the parties have executed an agreement specifically governing such use. Moreover, Würth Elektronik eiSos GmbH & Co. KG products are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. Würth Elektronik eiSos GmbH & Co. KG must be informed about the intent of such usage before the design-in stage. In addition, sufficient reliability evaluation checks for safety must be performed on every electronic component, which is used in electrical circuits that require high safety and reliability function or performance. By using Würth Elektronik eiSos GmbH & Co. KG products, the customer agrees to these terms and conditions.

## 10 License terms

This License Terms will take effect upon the purchase and usage of the Würth Elektronik eiSos GmbH & Co. KG wireless connectivity products. You hereby agree that this license terms is applicable to the product and the incorporated software, firmware and source codes (collectively, "Software") made available by Würth Elektronik eiSos in any form, including but not limited to binary, executable or source code form.

The software included in any Würth Elektronik eiSos wireless connectivity product is purchased to you on the condition that you accept the terms and conditions of this license terms. You agree to comply with all provisions under this license terms.

### 10.1 Limited license

Würth Elektronik eiSos hereby grants you a limited, non-exclusive, non-transferable and royalty-free license to use the software and under the conditions that will be set forth in this license terms. You are free to use the provided Software only in connection with one of the products from Würth Elektronik eiSos to the extent described in this license terms. You are entitled to change or alter the source code for the sole purpose of creating an application embedding the Würth Elektronik eiSos wireless connectivity product. The transfer of the source code to third parties is allowed to the sole extent that the source code is used by such third parties in connection with our product or another hardware provided by Würth Elektronik eiSos under strict adherence of this license terms. Würth Elektronik eiSos will not assume any liability for the usage of the incorporated software and the source code. You are not entitled to transfer the source code in any form to third parties without prior written consent of Würth Elektronik eiSos.

You are not allowed to reproduce, translate, reverse engineer, decompile, disassemble or create derivative works of the incorporated Software and the source code in whole or in part. No more extensive rights to use and exploit the products are granted to you.

### 10.2 Usage and obligations

The responsibility for the applicability and use of the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in a particular customer design is always solely within the authority of the customer. Due to this fact, it is up to you to evaluate and investigate, where appropriate, and to decide whether the device with the specific product characteristics described in the product specification is valid and suitable for your respective application or not.

You are responsible for using the Würth Elektronik eiSos wireless connectivity product with the incorporated Firmware in compliance with all applicable product liability and product safety laws. You acknowledge to minimize the risk of loss and harm to individuals and bear the risk for failure leading to personal injury or death due to your usage of the product.

Würth Elektronik eiSos' products with the incorporated Firmware are not authorized for use in safety-critical applications, or where a failure of the product is reasonably expected to cause severe personal injury or death. Moreover, Würth Elektronik eiSos' products with the incorporated Firmware are neither designed nor intended for use in areas such as military, aerospace, aviation, nuclear control, submarine, transportation (automotive control, train control, ship control), transportation signal, disaster prevention, medical, public information network etc. You



shall inform Würth Elektronik eiSos about the intent of such usage before design-in stage. In certain customer applications requiring a very high level of safety and in which the malfunction or failure of an electronic component could endanger human life or health, you must ensure to have all necessary expertise in the safety and regulatory ramifications of your applications. You acknowledge and agree that you are solely responsible for all legal, regulatory and safety-related requirements concerning your products and any use of Würth Elektronik eiSos' products with the incorporated Firmware in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Würth Elektronik eiSos. **YOU SHALL INDEMNIFY WÜRTH ELEKTRONIK EISOS AGAINST ANY DAMAGES ARISING OUT OF THE USE OF WÜRTH ELEKTRONIK EISOS' PRODUCTS WITH THE INCORPORATED FIRMWARE IN SUCH SAFETY-CRITICAL APPLICATIONS.**

### **10.3 Ownership**

The incorporated Firmware created by Würth Elektronik eiSos is and will remain the exclusive property of Würth Elektronik eiSos.

### **10.4 Firmware update(s)**

You have the opportunity to request the current and actual Firmware for a bought wireless connectivity Product within the time of warranty. However, Würth Elektronik eiSos has no obligation to update a modules firmware in their production facilities, but can offer this as a service on request. The upload of firmware updates falls within your responsibility, e.g. via ACC or another software for firmware updates. Firmware updates will not be communicated automatically. It is within your responsibility to check the current version of a firmware in the latest version of the product manual on our website. The revision table in the product manual provides all necessary information about firmware updates. There is no right to be provided with binary files, so called "Firmware images", those could be flashed through JTAG, SWD, Spi-Bi-Wire, SPI or similar interfaces.

### **10.5 Disclaimer of warranty**

THE FIRMWARE IS PROVIDED "AS IS". YOU ACKNOWLEDGE THAT WÜRTH ELEKTRONIK EISOS MAKES NO REPRESENTATIONS AND WARRANTIES OF ANY KIND RELATED TO, BUT NOT LIMITED TO THE NON-INFRINGEMENT OF THIRD PARTIES' INTELLECTUAL PROPERTY RIGHTS OR THE MERCHANTABILITY OR FITNESS FOR YOUR INTENDED PURPOSE OR USAGE. WÜRTH ELEKTRONIK EISOS DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT RELATING TO ANY COMBINATION, MACHINE, OR PROCESS IN WHICH THE WÜRTH ELEKTRONIK EISOS' PRODUCT WITH THE INCORPORATED FIRMWARE IS USED. INFORMATION PUBLISHED BY WÜRTH ELEKTRONIK EISOS REGARDING THIRD-PARTY PRODUCTS OR SERVICES DOES NOT CONSTITUTE A LICENSE FROM WÜRTH ELEKTRONIK EISOS TO USE SUCH PRODUCTS OR SERVICES OR A WARRANTY OR ENDORSEMENT THEREOF.



## **10.6 Limitation of liability**

Any liability not expressly provided by Würth Elektronik eiSos shall be disclaimed.

You agree to hold us harmless from any third-party claims related to your usage of the Würth Elektronik eiSos' products with the incorporated Firmware, software and source code. Würth Elektronik eiSos disclaims any liability for any alteration, development created by you or your customers as well as for any combination with other products.

## **10.7 Applicable law and jurisdiction**

Applicable law to this license terms shall be the laws of the Federal Republic of Germany. Any dispute, claim or controversy arising out of or relating to this license terms shall be resolved and finally settled by the court competent for the location of Würth Elektronik eiSos' registered office.

## **10.8 Severability clause**

If a provision of this license terms is or becomes invalid, unenforceable or null and void, this shall not affect the remaining provisions of the terms. The parties shall replace any such provisions with new valid provisions that most closely approximate the purpose of the terms.

## **10.9 Miscellaneous**

Würth Elektronik eiSos reserves the right at any time to change this terms at its own discretion. It is your responsibility to check at Würth Elektronik eiSos homepage for any updates. Your continued usage of the products will be deemed as the acceptance of the change.

We recommend you to be updated about the status of new firmware and software, which is available on our website or in our data sheet and manual, and to implement new software in your device where appropriate.

By ordering a wireless connectivity product, you accept this license terms in all terms.

## List of Figures

1	IoT application stack . . . . .	5
2	Components of a data platform . . . . .	6
3	System architecture . . . . .	7
4	System design . . . . .	8
5	AWS example . . . . .	10
6	Publish/Subscribe mechanism . . . . .	11
7	AWS IoT Core Microservice . . . . .	15
8	IoT Core: Things . . . . .	15
9	IoT Core: Create single Thing . . . . .	16
10	IoT Core: Specify thing properties . . . . .	16
11	IoT Core: Device Shadow . . . . .	17
12	IoT Core: Configure Device Certificates . . . . .	17
13	IoT Core: Download Certificates and Keys . . . . .	18
14	IoT Core: Create policies . . . . .	19
15	IoT Core: Select Policy to Attach with Certificate . . . . .	19
16	IoT Core: Attach Policy to Certificate . . . . .	20
17	IoT Core: MQTT Test Client . . . . .	21
18	IoT Core: Subscribe To Topic . . . . .	21
19	IoT Core: Publish data . . . . .	21

**Contact**

Würth Elektronik eiSos GmbH & Co. KG  
Division Wireless Connectivity & Sensors

Max-Eyth-Straße 1  
74638 Waldenburg  
Germany

Tel.: +49 651 99355-0  
Fax.: +49 651 99355-69  
[www.we-online.com/wireless-connectivity](http://www.we-online.com/wireless-connectivity)

**WÜRTH ELEKTRONIK** MORE THAN YOU EXPECT