SINGLE CHANNEL UHF RFID READER

Communication Specification Rev. 1

Model: TU-05-C01

9th Aug 2024

Contents

Overview 3
MQTT Communication 3
Device and Peripheral 3
Properties Values Category/Type 4
Attribute 4
Telematics 4
RPC (Remote procedure call) Request 4
RPC (Remote procedure call) Reply 4
Communication with the device through MQTT Broker 5
MQTT Server Setup 6
MQTT Topic 6
Overview on function of each topic 7
Attribute Topic (Device Publish System Attribute) 7
Attribute Topic (Device Update System Attribute) 7
Telematics Topic 7
RPC Request Topic 7
RPC Reply Topic 7
Communication Spam Protection 7
JSON Messages on Each Topic 8
Initial device connection after power up 9
Example of the system attribute data packet for the peripheral 10
Last Will Message on Device Publish System Attribute Topic (Device disconnected) 10
Further information on MQTT Topic Communications 11
Update Device's System Attribute 11
Telematics Data 12
RPC Request and RPC Reply 13
Communicating Through USB Serial Port 15
Setting for Serial Port 15
MQTT Topic Conversion 15
Other Differences on MQTT Topic vs UART communication 18
The Device Peripheral – Core System Properties 19
System Peripheral 19

System Only Attribute 19
Share Attribute 20
RPC Call and Response 20
Limitation and Counter Measure of MQTT server 21
The Device Peripheral – RFID Reader Properties 22
RFID Reader Peripheral 22
Attribute 22
Telematics Data 29
RPC Call and Response 31
RFID Tag Memory Map Layout 39
Tag Selection/Singulation 40
Example 40
Tag Locking and Tag Password 41
Gen2 Tag Passwords 41
Lock Mask and Lock Action 41
Tag Locking State Chart 42
Tag Locking process. 42
Tag Reading Process 43
TAG Reading/Writing Return Code 44

Overview

MQTT Communication

The device will communicate with the Host Server (backend Device Management Server) though an MQTT Broker. For more information on the MQTT Broker working concept, please refer to the link below,

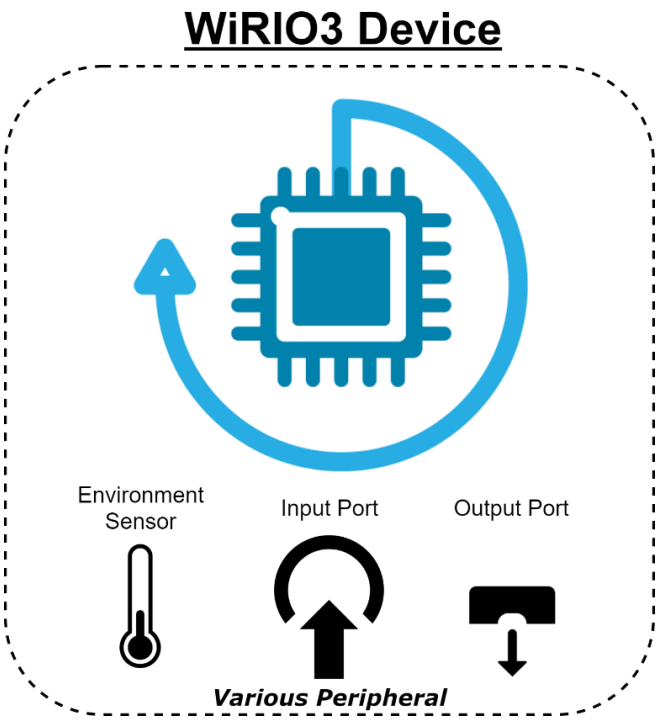
https://www.hivemq.com/mqtt-essentials/

https://www.youtube.com/watch?v=z4r4hIZcp40

Device and Peripheral

Each WiRIO3 Device can contain multiple types of Peripherals, e.g., Input Port, Output Port, Environment Sensor, etc... The host can request for the device properties which also include the Peripheral availability in the device any time by sending a blank JSON packet "{}" to the device.

Each Peripheral will have a Peripheral "cmdkey" included as part of the JSON key. This JSON key is used as the reference name that allows the Host Server to direct send and receive RPC or Telematics info to/from the peripheral.



Properties Values Category/Type

The properties category available are as follows,

- Attribute
- Telematics
- RPC (Remote procedure call) Request.
- RPC (Remote procedure call) Response.

Attribute

Attribute is the system properties and peripheral configuration setting in the device. The Attribute is divided into 2 types.

- Device Only Attribute
- Share Attribute

Device Only Attribute

This type of attribute is only allowing the device to modify and update to the Host Server. Host Server **does not** allow to modify the value of the attribute.

The device will publish any changes on attribute in the Device Publish System Attribute topic.

Share Attribute

Both the device and the host allow to modify the attribute value.

Telematics

Telematics is the sensor or I/O port status that publishes out to the host when the status changes. The Host Server will monitor and process this real time status value.

RPC (Remote procedure call) Request

RPC Request is sent to the device when the Host Server wanted to make changes on sensor or Outport port status value.

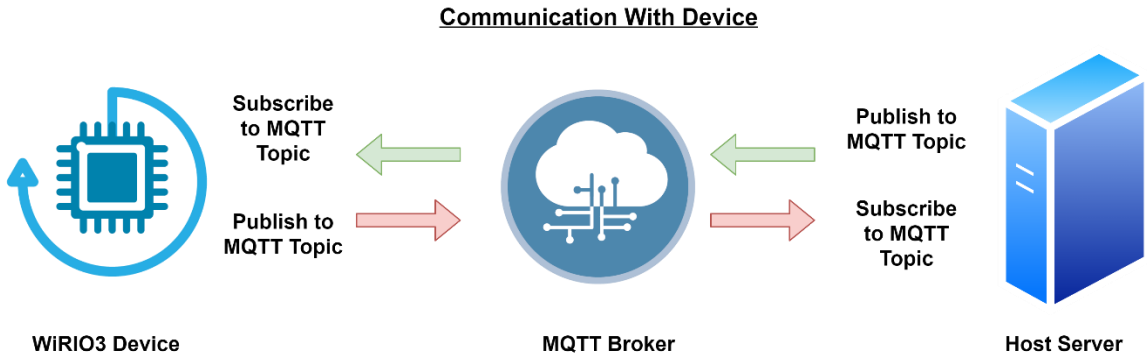
RPC (Remote procedure call) Reply

After the devices receive an RPC Request, it will process the request and return result in RPC Reply.

Communication with the device through MQTT Broker

The WiRIO3 Device and Host Server (backend Device Management Server) will connect to a MQTT Broker. The device will subscribe to the required topic to listen to any message published or request by the Host Server and the device will publish any sensor Telematics information into the pre-defined MQTT topic.

The Host Server requires to subscribe to the device publish pre-defined MQTT topic to listen to the device's published message and process the sensor data accordingly.



MQTT Server Setup

MQTT Broker Connection Type	MQTT TCP/Web-Sock connection with JSON as data packet format
TCP Port Number	User configurable
Encryption	Public CA or Private Certificate, TLS/SSL
Login Control	Available

** The server address and port number can change according to the user requirement using the WiRIO3 Device Configuration Android Apps.

The supported MQTT Topic format are,

- WiRIO3 Topic Format
- Thingsboard.io Topic Format
- User Customizable Topic Format

The selection and configuration of the MQTT Topic by using the WiRIO3 Device Configuration Android Apps.

MQTT Topic

In each of the MQTT Topic formats selected, there is a total of 5 MQTT Topic either for device to publish messages to the server or for device to subscribe to the topic to listen to any messages from the server.

The 5 MQTT Topic required by the device are as follow,

Type of Topic	*Properties Type*	*Device*	*Host Server*
Telematics Data	Telematics	Publish	Subscribe
Device Publish System Attribute	Attribute	Publish	Subscribe
Update Device's System Attribute	Attribute	Subscribe	Publish
RPC Request	RPC Request	Subscribe	Publish
RPC Reply	RPC Reply	Publish	Subscribe

The default topic setting is as below,

Type of Topic	*WiRIO3 Default*	*Thingsboard.io Default*
Telematics Data	W3/<DeviceID>/telemetry	v1/devices/me/telemetry
Device Publish System Attribute	W3/<DeviceID>/attributes	v1/devices/me/attributes
Update Device's System Attribute	W3/<DeviceID>/attributes	v1/devices/me/attributes
RPC Request	W3/<DeviceID>/rpc/request/<RequestID>	v1/devices/me/rpc/request/<RequestID>
RPC Reply	W3/<DeviceID>/rpc/response/+	v1/devices/me/rpc/response/+

The <DeviceID> is the unique ID of the device in the format of WIRIO3_AABBCCDDEEFF where AABBCCDDEEFF is the 6 bytes hex string in upper case. E.g., WIRIO3_43DF01543AF0.

The <RequestID> is a reference number provided by the Host Server and echo back to the Host Server during the RPC Reply (Attach the end of the RPC Reply topic).

The "+" wildcard on the RPC reply to topic will allow the subscriber to receive any topic with <RequestID> value attached at the end of the topic. For more details on the MQTT topic wildcard, please refer to the Standard MQTT Broker documentation.

Overview on function of each topic

Attribute Topic (Device Publish System Attribute)

The device will publish any system and peripheral properties setting in this topic. During initial connection to the MQTT Broker, the device will auto publish all the system and peripheral properties through this topic.

Attribute Topic (Device Update System Attribute)

The device will subscribe to this topic and monitor any request from the Host Server to update the **share** attributes. If the Host Server requires modifications to modify the Share Attribute, the Host Server will publish the request into this topic.

Telematics Topic

The device will use this topic to push the device's telematics data. The Host Server should monitor this topic for any new sensor information update from the device and process the sensor data accordingly.

RPC Request Topic

If the server requires the device to do certain action, the Host Server will publish the request into this topic. On the request message's topic, Host Server required to publish the message into the topic with a <RequestID> append at the end of the topic, e.g., W3/<DeviceID>/request/<RequestID>. During the RPC reply, the <RequestID> will be append at the end of the reply to message's topic. The device **will not** validate/check on the <RequestID> value.

Please note that the <RequestID> is compulsory. The system can provide any arbitrary number when publishing the RPC Request.

E.g.

W3/C45BBE821F38/request/48

RPC Reply Topic

Once the device receives and processes the RPC request from the server, the device will reply to the server through this topic. The <RequestID> on the request message's topic, will be append to the reply to message's topic. Depending on the request, some of the request will also cause sensor value to publish in Telematics Topic.

Communication Spam Protection

On all the device listening topics (Device Update System Attribute Topic and RPC Request Topic), in combine, the timing between each message must have a minimum gap period of 150ms. Any data packet that has a timing gap period that is less than this, will be ignore.

JSON Messages on Each Topic

On the **all the packet sending out from the device**, it will have minimum number of keys as below,

```
{
  "deviceid": "WIRIO3_C45BBE821F38",
  "rssi": -62,
  "pktno": 1598742192,
  ...
  ...
}
```

On **all the packet sending to the device** from the back-end server, the minimum number of keys require are the "deviceid" key,

```
{
  "deviceid": "WIRIO3_43DF01543AF0",
  ...
  ...
}
```

If the "deviceid" do not match the ID of the device, the device will ignore the message.

But if the MQTT Topic format selection is either WiRIO3 format, Thingsboard.io or UART communication format, the key "deviceid" is **NOT** required and will be ignore by the device.

Others JSON key that the device published out are as below,

Key	**Description**
deviceid	The unique identifier for each device
sec	** Current Epoch times in second. (Only available in Telematics Topic)
rssi	Device Wi-Fi Received Signal Strength Index (Only Available if device is on Wi-Fi Link)
pktno	A running number increase by 1 when device **publish** a packet on any of the device publish topic

** The EPOCH time "sec", only available (in Telematics Topic) if the device is able to obtain the real time clock through the NTP server or Host Server has Updated the "s.sys.epochsec" key and "d.sys.epochvalid" is True.

The detail NTP server setting and configuration is available in the WiRIO3 Device Configuration Android Apps.

Initial device connection after power up

During the initial connection, it will publish out the all the available system and peripheral attribute.

The property key listed above starting with “d.xxx.xxx” is Device Only Attribute and not modifiable by the Host Server. If there are any changes on the system attribute, the device will publish it again.

The property key with “s.xxx.xxx” is Share Attribute that can be modify by either device or Host Server (Refer to section “Device Update System Attribute Topic” for further information).

If the device has any peripheral configurable value, the system will publish it together with the above attribute. Below is the example of the Device Published Attribute.


```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "rssi": -72,
  "pktno": 2,
  "d.sys.linkup": true,
  "d.sys.addr": "192.168.0.101",
  "d.sys.iface": "WIFISTA",
  "d.sys.ssid": "My_AP",
  "d.sys.fwid": "0301",
  "d.sys.datecode": "210516",
  "d.sys.model": "M5-01-01",
  "d.sys.name": "Enviroment Sensors Data Logger",
  "d.sys.desc": "Temp-Humidity data Logging in sdcard",
  "d.sys.perip": [
    {
      "cmdkey": "envsensor",
      "feature": "tempc|hum"
    }
  ],
  "d.sys.epochvalid": true,
  "s.sys.epochsec": 1629021412
}
```

Example of the system attribute data packet for the peripheral.

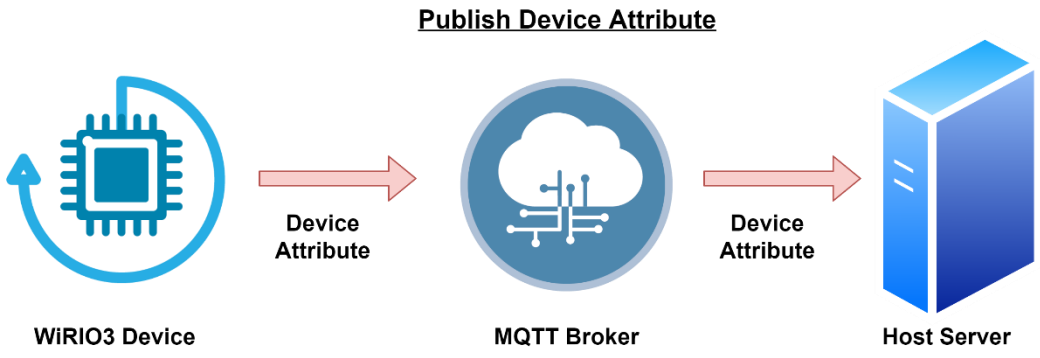
```
"d.sys.perip": [
  {
    "cmdkey": "uhfrfid",
    "feature": "NA"
  },
  {
    "cmdkey": "outputport",
    "feature": "4ch"
  }
],
```

Example of a device d.sys.perip attribute indicating that the device has more than one Peripheral.

Last Will Message on Device Publish System Attribute Topic (Device disconnected)

During device initial connection to the MQTT Broker, it will register a Last Will message that will allow the MQTT Broker to publish out when the device disconnected. The JSON message for the Last Will Message is as below,

```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "d.sys.linkup": false
}
```



Further information on MQTT Topic Communications

Update Device's System Attribute

If there is any value that allows the Host Server to modify including the device peripheral setting, the Host Server can publish the key to modify it in this topic.

E.g., Change the UHF reader transmission power.

```
{
  "deviceid": "WIRIO3_43DF01543AF0",
  "s.uhfrfid.power": 28
}
```

Example of the device response.

```
{
  "deviceid": "WIRIO3_43DF01543AF0",
  "rssi": -60,
  "pktno": 324,
  "s.uhfrfid.power": 28
}
```

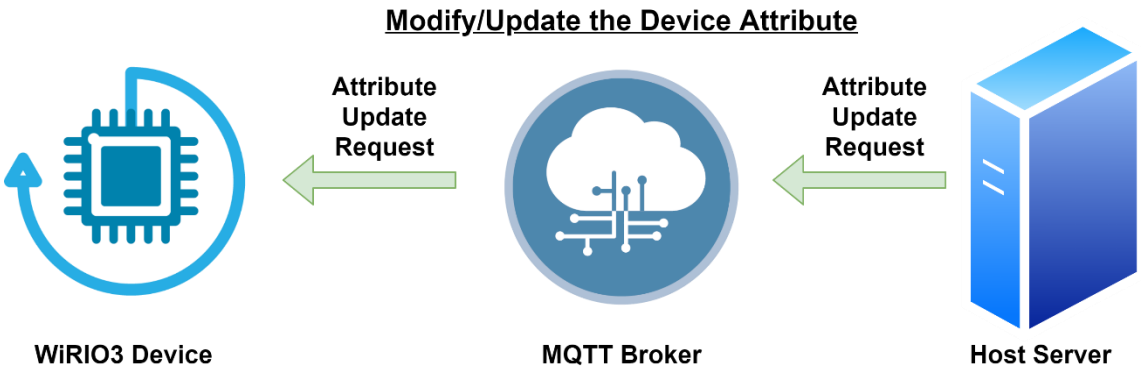
If the Host Server send a blank JSON with only the "deviceid" key (for customized MQTT Topic), or total blank JSON packet (for WiRIO3 and Thingsboard Topic) the device will publish out all the device available attribute in the Device Publish System Attribute Topic.

```
{
  "deviceid": "WIRIO3_43DF01543AF0"
}
```

E.g. Blank request by the host for user customize MQTT Topic.

```
{}
```

E.g., Blank request by the host for WiRIO3 or Thingsboard.io MQTT Topic.

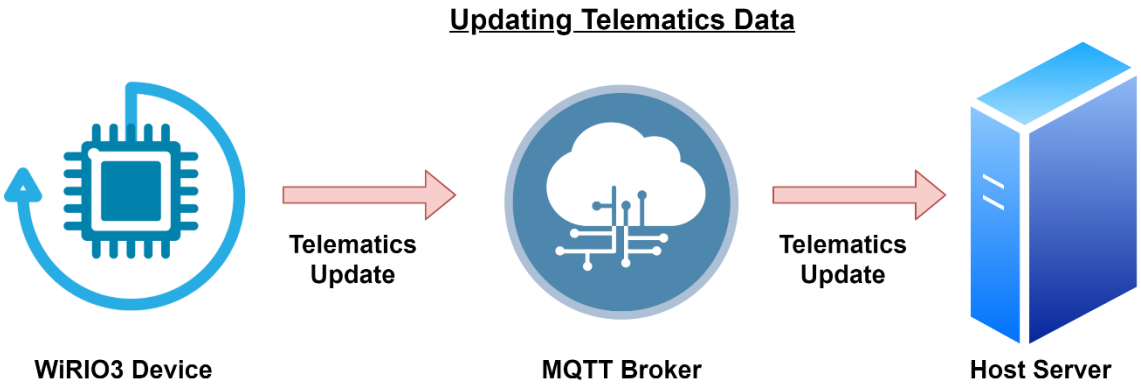


Telematics Data

The device will use this topic to publish any telematics data detected by the host. On the detail key/value on the telematics data, please refer to the individual device communication specification for more information.

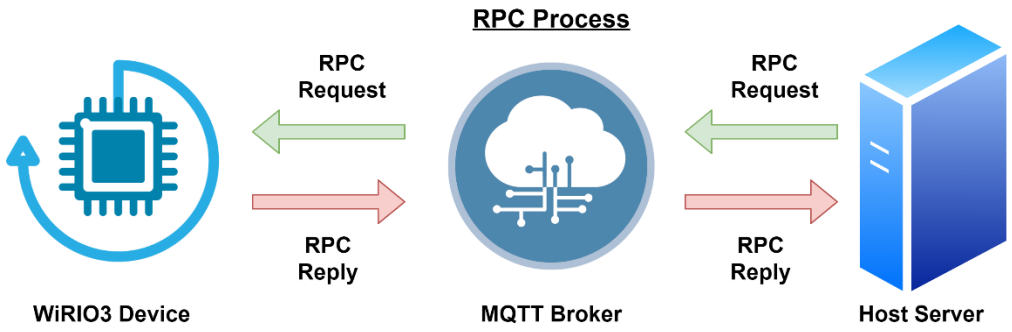
```
{
  "deviceid": "WIRIO3_43DF01543AF0",
  "rssi": -72,
  "pktno": 35,
  "sec": 1629104885,
  "t.envsensor.param": [
    {
      "tempc": "31.5C",
      "hum": "65.3%"
    }
  ]
}
```

E.g., Above is a Temperature/Humidity Sensor device publish out the latest temperature and humidity reading.



RPC Request and RPC Reply

The Host Server can request action on the device's peripheral that has RPC feature. E.g. Output Port Peripheral.



The host requested to set the output port on channel 0 and channel 2 in RPC request topic,

e.g.

W3/ WIRIO3_43DF01547AB1/rpc/request/33

or

W3/ WIRIO3_43DF01547AB1/rpc/request/ (if the <RequestID> is not required).

```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "r.outputport.param": [
    {
      "ch": 0,
      "seton": true
    },
    {
      "ch": 2,
      "seton": false
    }
  ]
}
```

Once the device has processed the RPC request, it will reply to the request in the RPC Response topic, e.g.

W3/ WIRIO3_43DF01547AB1/rpc/response/33

```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "rssi": -72,
  "pktno": 102,
  "r.outputport.result": true
}
```


If the RCP request process successfully, it will return the key/value,

"r.<cmdkey>.result" : true

Else it will return false.

"r.<cmdkey>.result" : false

In example above, the <cmdkey> is "outputport".

Please take note that the RPC request is a **blocking process**. The device only handles the subsequent request after the RPC Reply is published out. The device would reply "false" directly if the request received before the device publishes the reply message for the previous request.

Communicating Through USB Serial Port

For the hardware that supports serial port communication, users have the option to communicate with the device through USB serial port while maintaining the JSON communication format structure.

Users can use either Serial Port **OR** MQTT Broker as the JSON communication transport device. Selection of communication mode is set using the WiRIO3 Device Configuration Android Apps.

Users can only select either using the MQTT Broker **OR** using Serial Port (USB Virtual Comm. Port) for communication. The MQTT Broker will take precedent and disable the Serial Port Communication during device boot up if both interfaces are enabled.

Setting for Serial Port

Item	*Setting*
Baud Rate	921600
Bit Number	8
Number of Parity Bit	None
Number of Stop Bit	1

MQTT Topic Conversion

Due to serial communication having only one communication channel, all the topics involved in the MQTT Communication are converted into JSON object key in the JSON communication packet. An extra outer JSON bracket will encapsulate the original JSON message.

List of the MQTT topic conversion to the outer JSON bracket as below,

MQTT Topic	*Convert to JSON Key*	*Direction*
Telematics Data	telematics	From Device
Device Publish System Attribute	attribute	From Device
Update Device's System Attribute	attribute	From Host/Server
RPC Request	rpcreq	From Host/Server
RPC Reply	rpcreply	From Device

Example of the RPC Request and RPC Reply,

```
{
  "rpcreq": {
    "r.uhfrfid.start": 1
  }
}
```

```
{
  "rpcreply": {
    "deviceid": "WIRIO3_943CC699C7F0",
    "pktno": 885168941,
    "r.uhfrfid.result": true
  }
}
```

Please noted that an extra outer JSON bracket with key "rpcreq" will encapsulate the original MQTT message.

Example of the telematics packet received from the device are as below,


```
{
  "telematics": {
    "deviceid": "WIRIO3_943CC699C7F0",
    "pktno": 4097455911,
    "t.uhfrfid.param": [
      {
        "ts": "506015",
        "ch": "4",
        "rssi": "-68",
        "tag": "E200001D520D02222100C46F"
      },
      {
        "ts": "506215",
        "ch": "4",
        "rssi": "-69",
        "tag": "E200001D520D01481970783E"
      },
      {
        "ts": "506222",
        "ch": "4",
        "rssi": "-63",
        "tag": "E2801191A50300608C8C4F82"
      }
    ]
  }
}
```


Please noted that an extra outer JSON bracket with key “telematics” will encapsulate the original MQTT message.

Example of Attribute request,

```
{
  "attribute": {}
}
```

And the Attribute reply.


```
{
  "attribute": {
    "deviceid": "WIRIO3_943CC699C7F0",
    "pktno": 885168943,
    "d.sys.linkup": true,
    "d.sys.addr": "0.0.0.0",
    "d.sys.iface": "NONE",
    "d.sys.ssid": "",
    "d.sys.fwid": "0501-EM106R2",
    "d.sys.datecode": "210910",
    "d.sys.model": "TU-02-U",
    "d.sys.name": "Cabinet Tag Reader",
    "d.sys.desc": "Multi Tag Reader for Cabinet",
    "d.sys.perip": [
      {
        "cmdkey": "uhfrfid",
        "feature": "NA"
      }
    ],
    "d.sys.epochvalid": false,
    "d.uhfrfid.ch": 4,
    "d.uhfrfid.pwrmax": 28,
    "d.uhfrfid.pwrmin": 5,
    "d.uhfrfid.csmax": 100,
    "s.uhfrfid.mode": 0,
    "s.uhfrfid.q": 4,
    "s.uhfrfid.auto": 0,
    "s.uhfrfid.period": 10,
    "s.uhfrfid.power": 28,
    "s.uhfrfid.cacheperiod": 1000,
    "s.uhfrfid.cachesize": 100,
    "s.uhfrfid.semode": 0
  }
}
```


Please noted that an extra outer JSON bracket with key “attribute” will encapsulate the original MQTT message.

Other Differences on MQTT Topic vs UART communication

The other difference between the MQTT Broker vs serial UART port communication beside the MQTT Topic, are as below,

- <RequestID> does not supported in Serial Communication
- JSON key “rssi” is not available in Serial Communication
- Attribute related to MQTT server or Wi-Fi/Ethernet interface is not relevant, please ignore it.
- JSON key “d.sys.linkup” will always “true”.

The Device Peripheral – Core System Properties

System Peripheral

Key	Value
keycmd	sys

System Only Attribute

Key	Value Type	Description
d.sys.model	String	Device Model (TU-04-C04)
d.sys.name	String	Device Name (WiRIO3 IoT Reader)
d.sys.desc	String	Device Description (IoT 4ch UHF RFID Tag Reader)
d.sys.fwid	String	Firmware ID (E.g., 0507-04-EM109R1)

Key	Description
d.sys.addr	Device current IP address
d.sys.iface	Device connected interface, either “WiFISTA” for Wi-Fi Link Interface or “ETHERNET” for Ethernet Link Interface
d.sys.ssid	Device Wi-Fi Received Signal Strength Index (Only Available if device is on Wi-Fi Link)
d.sys.fwid	Device Firmware ID (unique for each type of device)
d.sys.datecode	Firmware version Date Code in YYMMDD
d.sys.model	Device model number
d.sys.name	Device Name
d.sys.desc	Device description
d.sys.perip	Available device peripheral in JSON Array. Data in this key will provide detailed properties of the available peripheral in the device. Available keys under d.sys.perip are as below, cmdkey : The peripheral reference ID. feature : Value that describes the feature of this peripheral.
d.sys.linkup	True of False to indicate if the device connected or not connected to the MQTT Broker server.
d.sys.epochvalid	True or false to indicate if the EPOCH time is valid or not valid.
d.sys.rpcbusy	Device is busy processing RPC requests.

Share Attribute

Key	Description
s.sys.epochsec	EPOCH time in second. Update the system Real time through this key.
s.sys.mqttsenddelay	Minimum delay period between 2 consecutive MQTT Packet sending. This is to prevent packet sending too fast causing MQTT brokers to drop packets. Value 100~1000 (millisecond).

RPC Call and Response

Key	Value Type	Description
r.sys.reboot	Boolean	Reboot the device

Limitation and Counter Measure of MQTT server

Due to the nature of the TCP communication protocol used by the MQTT server, and the turnaround time required by the MQTT server to process an incoming TCP packet, a minimum delay period between sending two consecutive TCP packets can be set.

User can configure the delay period from 100~1000ms by setting the attribute "s.sys.mqttsenddelay".

If the device generates multiple JSON messages between the sending period, the system will cache the JSON messages in the device memory until the next sending time slot. The system will combine all the cache JSON messages and send out the combined message in one TCP packet.

Backend Application Server that monitors the topic will receive multiple JSON messages in one big chuck.

For the Application Server to properly Deserialize the JSON messages, the received message must first go through a splitter module to split out the JSON messages one by one and Deserialize it accordingly.

Example Telemetry TCP packet received with multiple JSON messages,

```
{ "deviceid":…,"rssi":.., pktno":335885456, …}{ "deviceid":…,"rssi":..,
"pktno":335885457, …}{ "deviceid":…,"rssi":..,
"pktno":335885458, …}{ "deviceid":…,"rssi":.., "pktno":335885459, …}
```

Splitter Module to split out the JSON messages and process it one by one.

```
{ "deviceid":…,"rssi":.., "pktno":335885456, …}

{ "deviceid":…,"rssi":.., "pktno":335885457, …}

{ "deviceid":…,"rssi":.., "pktno":335885458, …}

{ "deviceid":…,"rssi":.., "pktno":335885459, …}
```

The Device Peripheral – RFID Reader Properties

RFID Reader Peripheral

Key	Value
keycmd	uhfrfid
feature	NA

Attribute

Device Only Attribute

Key	Value Type	Description
d.uhfrfid.ch	Integer	Indicate number of available RFID Channel
d.uhfrfid.pwrmax	Integer	Maximum reader power in dBm
d.uhfrfid.pwrmin	Integer	Minimum reader power in dBm
d.uhfrfid.csmax	Integer	Maximum Tag Cache Memory size (in number of tag)
d.uhfrfid.antstate	Array of Boolean	Each element indicates the status on whether the antenna channel connected to an external antenna or not. Eg. [true,false,true,true], indicates that Antenna Channel 1,3,4 is connected with an antenna and channel 2 is NOT connected with an antenna.
d.uhfrfid.antison	Array of Boolean	Each element indicated if the antenna output channel is ON (true) or OFF (false). Eg. [true,false,true,true}, indicate that Antenna Output Channel 1,3,4 is ON and channel 2 OFF.
d.uhfrfid.temp	Integer	RFID reader internal temperature in degree Celsius. Temperature must not exceed 85°C. Once it reaches 85°C, it will force the reader to stop the tag detection to prevent the system from overheating.
d.uhfrfid.readerver	String	Internal RFID Reader Module Firmware ID

Share Attribute

The function of the Share Attribute can be categories as below,

- Tag Inventory and Cache Setting
- Tag Reading Mode Setting
- Reader Setting
- Demo and Testing

Tag Inventory and Cache Setting

Configuration of the reading behavior, timing and triggering during tag inventory process.

Key	Value Type	Description
s.uhfrfid.devreadperiod	Integer	Period in millisecond for the device to read for the tag in the multi tag reading loop. Once the tag reading is started, within the period, the reading process cannot be interrupted. The device will only return after finished the period. The setting range is 500ms to 10000ms.
s.uhfrfid.auto	Boolean	False: Will only start Tag Inventory/Reading upon request (r. uhfrfid.start=true) and stop when reach s.uhfrfid.period True: Will continue Tag Inventory/Reading when r.uhfrfid.start=true and stop when r.uhfrfid.start=false. Please note if it is set to True, the reader will start the Tag Inventory/Reading process immediately after the system is power up.
s.uhfrfid.period	Integer	Reading period in second when s. uhfrfid.auto=false. Value: 5sec ~ 300sec
s.uhfrfid.cachetagremove	Boolean	If true, will auto remove the tag from the cache memory after the period given by s.uhfrfid.cacheperiod. If false, the tag data will still be in the cache memory until the tag inventory process stops by calling r.uhfrfid.start=false or the reader stops when reading period is over during s.uhfrfid.auto=false. No more new tag will add into the cache once the cache memory is full.
s.uhfrfid.cacheperiod	Integer	The period for the detected tag to be in the Tag Cache Memory before removing it from the cache. After a tag is detected, if the tag is still staying in the reader reading range, the same tag will be re-detected again. If the re-detected period is less than the cache period, the tag will continue to stay in the cache memory.

		The system will only update the tag info in the Telematics topic when the tag detected is newly detected tag that is not already available in the Tag Cache Memory. The value provided is millisecond. Value=500ms to 60000ms This setting is only valid if s.uhfrfid.cachetagremove=true.
s.uhfrfid.tagremoveupd	Boolean	When set to True, it will send out the tag information when the tag is no more detected and remove from the cache memory.
s.uhfrfid.antChangeUpd	Boolean	Enable(true)/Disable(false). When enabled, it will send out the detected tag in Telematics if the tag is reading from different antenna channels. If it is disable, it will send out the detected tag info ONCE only during the initial tag detection that is not already available in the Tag Cache Memory irrespective of the of the signal coming from which antenna channel.
s.uhfrfid.enbforceupd	Boolean	When set to true, the system will send out the tag info that is currently in the cache memory. The period is depend on the setting in s.uhfrfid.forceupdperiod
s.uhfrfid.forceupdperiod	Integer	Range 3~100Second = period in force update tag info in the cache memory. Only valid if s.uhfrfid.enbforceupd=true.

Tag Selection/Singulation Filter Criteria

While the system is reading/inventory the surround tag, the system can filter off the unwanted tag and only read the tag that matches the selection filter criteria.

Key	Value Type	Description
s.uhfrfid.selfilteroption	Integer 0~4	Tag Selection mode. 0 = Disable Tag Selection Filter feature. 1 = Tag Selection Filter with the Tag EPC data. (It will ignore key "s.uhfrfid.selfilteraddr" and start compare from the first bit of the Tag EPC data) 2 = Tag Selection Filter with data in TID Memory Bank 3 = Tag Selection Filter with data in USER Memory Bank 4 = Tag Selection Filter with data in EPC Memory Bank
s.uhfrfid.selfilteraddr	32bit Hex String	Define the starting bit address in the Tag Memory to compare with the s.uhfrfid.selfileterdata. If bit address is 0x20, it will have value "20". Please refer to RFID Tag Memory Map Layout.
s.uhfrfid.selfilterlenbit	Integer	Specified number of BIT to compare/match during tag filter selection
s.uhfrfid.selfilterdata	String of Hex Byte, Max 64byte	Define the data use for comparing against the data in the Tag memory according to filtering type configured in "s.uhfrfid.selfilteroption". The system will start to match/compare from the MSB bit for the first byte and total bit to compare is according to s.uhfrfid.selfilterlenbit.
s.ufhrfid.selfilterinvert	Boolean	If set to True, it will select the Tag that **NOT** matching the tag selection filter criteria.

For more information on the Tag Selection/Singulation filter criteria, please refer to section "Tag Selection Filter".

Auto Tag Data Reading

While the system is inventory /reading the surround tag, the system can read one of the memory banks in the tag to allow fast acquiring of the tag data.

Key	Value Type	Description
s.uhfrfid.enbtagdata	Boolean	Enable Tag Data reading during Inventory process. When enabled, the system will auto read the data from one of the memory banks (RESERVED, EPC, TID, USER).
s.uhfrfid.tagdatamembank	Integer 0~3	Part of Enable Tag Data Reading Setting. Specified the memory bank to read during inventory process, 0 = RESERVED 1 = EPC 2 = TID 3 = USER
s.uhfrfid.tagdatareadaddr	32bit Hex String	Part of Enable Tag Data Reading Setting. Set the start address (16bit addressing) of the tag memory to read. Eg. If Bit Address of the Tag Memory is 0x20, the s.uhfrfid.tagdatareadaddr will be set to “2”.
s.uhfrfid.tagdatawordcount	Integer 1~96	Part of Enable Tag Data Reading Setting. Set the total number of words (16bit) to read.
s.uhfrfid.enbtagpass	Boolean	Enable(true)/Disable(false) Tag password during tag inventory.
s.uhfrfid.tagpass	32Bit Hex String	32Bit Tag password in Hex String. Eg. “ff2d2050” will have 0xff,0x2d,0x20,0x50 as tag password.

When turning on the tag data reading, if the tag is locked with password, the s.uhfrfid.enbtagpass must be set to True with s.uhfrfid.tagpass has the tag password.

If the tag data is read successfully, it will return together with other tag information in the Telematics Data with the key “tagdata”. Please refer to the Telematics Data section t.uhfrfid.param for more information.

Tag Reading Mode Setting

Various reader reading tuning values are available to allow fast reading of the surrounding tag according to the various environment conditions and application requirements.

Key	**Value Type**	**Description**
s.uhfrfid.dynamicq	Boolean	When true (1), the Q factor is set to Dynamic Q and the s.uhfrfid.q setting will be ignore
s.uhfrfid.q	Integer	Tag Query Q factor 0~16 (default = 4), Higher Q value suitable high tag density detection but with slower return respond. If the factor is -1, the Dynamic Q factor is enabled.
s.uhfrfid.semode	Integer	Set the Tag Query Session mode to S0~S3. Value: 0 to 3
s.uhfrfid.target	Integer 0~3	0 = Target A. Read 'A' tag and move it to state 'B'. 1 = Target B. Read 'B' tag and move it to state 'A'. 2 =Target A-B. Reads 'A' tags one at a time and moves them into the 'B'. Once no more tag found, start to read 'B' tags one at a time and moves them into the 'A' state. The process will repeatedly until it finishes reading all the tags. 3 = Target B-A. Same as Target A-B but in reverse order.
s.uhfrfid.epcextended	Boolean	Enable(true)/Disable(false) EPC Extended Info. When enabled, the tag ID provided in the Telematics will have format as below, PC(2byte)+EPC+CRC(2byte) PC – Protocol Control Word EPC – Tag EPC Memory, word size is dependent on the PC word setting. CRC – Tag CRC Word calculate by the tag.

Reader Setting

Key	**Value Type**	**Description**
s.uhfrfid.power	Array of double	Set antenna output power in dBm, range 1dbm to 33dbm. E.g. [28], will set Antenna Channel 28dBm The power is in 1dBm step.

Demo and Testing

Key	**Value Type**	**Description**
s.uhfrfid.demo	Boolean	When set to true, all the detected tags will be immediately posted out in the Telemetry Topic. Internal cache will always clear.

Telematics Data

Key	Value Type	Description
t.uhfrfid.param	Array of object	Return the tag detected/undetected result. Each element of the array will contain key/value as below, • tag : UHF Tag EPC data in format <PC+EPC+CRC> display in hex string. • ch : Antenna Channel number where the tag is detected. The Channel number starts from 1. • rssi : is the tag data return signal strength indicator • ts : is the system time stamp since power up in millisecond. • state : integer 0,1,2. ○ 0 = the previously detected tag is no longer being available and has been remove from the tag cache memory. ○ 1 = tag is newly detected and add into the tag cache memory. ○ 2 = resend the tag info that is currently in the cache memory. • tagdata : return the tag data if s.uhfrfid.enbtagdata is enabled and the tag data is successfully read.
t.uhfrfid.readstarted	Boolean	True if the reading started. False if the reading has stopped.
t.uhfrfid.temp	Integer	RFID reader internal temperature in degree Celsius. Temperature must not exceed 85°C. Once it reaches 85°C, it will force the reader to stop the tag detection to prevent the system from overheating.
t.uhfrfid.temperr	Boolean	Set to true if the RFID reader internal temperature is 85C and above. (Over temperature error)
t.uhfrfid.antstate	Array of Boolean	It will send out when the antenna connected status changes. Each element indicates if the antenna channel has detected an external antenna connected to the antenna port. E.g. [true,false,true,true], indicates that Antenna Channel 1,3,4 is connected with an antenna and channel 2 is NOT connected with an antenna.
t.uhfrfid.antison	Array of Boolean	Will sent out when antenna connected status is changed. Each element indicated if the antenna output channel is ON (true) or OFF (false). Eg. [true,false,true,true}, indicate that Antenna Output Channel 1,3,4 is ON and channel 2 OFF.
t.uhfrfid.writetag	JSON Object	Return the Tag Writing request result code. The Object will contain key below, “writeoption”, The last request tag writing option “code”, The result code in 16bit Hex string, please refer to section “TAG Reading/Writing Return Code” for more info.

t.uhfrfid.readtag	JSON Object	Return the Tag Writing request result code. The Object will contain key below, “readoption”, The last request tag writing option “code”, The result code in 16bit Hex string. Please refer to section “TAG Reading/Writing Return Code” for more info.

Example of a telematics packet


```
{
  "deviceid": "WIRIO3_C45BBE821F38",
  "rssi": -66,
  "pktno": 1598766424,
  "sec": 1632176066,
  "t.uhfrfid.param": [
    {
      "ts": "34815396",
      "ch": "1",
      "rssi": "-66",
      "tag": "3000E200001D520D006019501E627259"
    },
    {
      "ts": "34815445",
      "ch": "1",
      "rssi": "-71",
      "tag": "3000E200001D5211017904309E717E4F"
    },
    {
      "ts": "34815445",
      "ch": "1",
      "rssi": "-62",
      "tag": "3000E200001D520D02222100C46F7057"
    }
  ]
}
```

RPC Call and Response

Tag Inventory/Auto reading

Key	**Value Type**	**Description**
r.uhfrfid.start	Boolean	True • Start the reading process. • If s.uhfrfid.auto=true, it stop reading when it reach the s.uhfrfid.period setting. • Clear the Tag Cache memory to allow new tag info to update in the telematics topic. False • Immediately stop detecting the card. Return result in RPC Call Response r.uhfrfid.result = true if the command process successfully.
r.uhfrfid.reload	Boolean	Will flush the internal Tag Memory Cache and rescan all the tag. Set to True only. This command only allows when the reader is currently reading tag.

Example of the RPC Request,

```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "r.uhfrfid.start":0
}
```

Example of the RPC Reply

Below is the respond in RPC Reply topic once the reader has started the Tag Inventory/Reading process,

```
{
  "deviceid": "WIRIO3_43DF01547AB1",
  "rssi": -72,
  "pktno": 102,
  "r.uhfrfid.result": true
}
```

Tag Writing

The RFID Tag writing include,

- Writing on one tag at a time.
- Optional tag selection filter according to data in any memory bank.
- Writing on tag memory bank RESERVED, EPC, TID and USER with user definable writing data size and address location in the tag memory.
- Tag writing data must be in multiple of 16bit.
- Set the Tag Locking flag on an individual memory bank.
- Kill the RFID Tag and prevent the tag from further reading. Killed tag will not respond to any future RFID reading or writing request.

All the Tag Writing request will be put under key value “r.uhfrfid.writetag” JSON object. The Tag Users can request for writing when the reader is idling or while the reader is on Tag Inventory/Reading mode.

If the Tag Writing request command fulfils the parameter requirement and accepts for processing, it will respond with result = TRUE in RPC Reply Topic. The request will be queue for processing. Once the request is processed, the processed result will be returned in Telematics Topic under key “t.uhfrfid.writetag”.

If the system is currently busy or the request command provided does not fulfill the parameter requirement, it will respond with result = FALSE.

For more information on the Tag Writing telematics result, please refer to “t.uhfrfid.writetag” under “Telematics Data” Section.

Key	**Value Type**	**Description**
r.uhfrfid.writetag	JSON Object	All the parameters required will be place under this JSON object.

Parameter in key “r.uhfrfid.writetag” JSON object

Key	**Value Type**	**Description**
writeoption	Integer 0~6	Tag Writing process available option are, 0 = Write to RESERVED memory bank 1 = Write to EPC memory bank 2 = Write TID memory bank 3 = Write USER memory bank 4 = Replace Tag EPC (will directly replace existing EPC and update PC and CRC in the tag) 5 = Lock Tag memory 6 = Kill Tag
timeout	Integer, 16bit	Optional. Tag writing time out in millisecond. Default is set to 1000ms.

tagselection	JSON object	Optional. JSON object that provides the tag selection/filter criteria when writing the tag. If not provided, it will write to all the surrounding detected tag. For further detail, please refer to the Parameter in key "tagselection" JSON object at table below.
writeaddr	32bit Hex String	Set the start address (16bit addressing) of the tag memory to write. E.g., If Bit Address of the Tag Memory is 0x20, the writeaddr will be set to "2". Please refer to RFID Tag Memory Map Layout for the bit addressing. Require by writeoption 0, 1, 2, 3.
writedata	String of Hex Byte Max 32 words	Data byte to write on the tag, require by writeoption 0, 1, 2, 3. Must be in 16bit words for writeoption 0,1,2,3. Max 32 words. On writeoption 0, 1, 2, 3, The Tag writing will be in multiple of 16bit words. The Hex string provided will be auto zero padded to fulfill the requirement.
lockmask	16bit Hex String	Lock tag Mask Word Setting Require by writeoption=5 only. Refer to Section "Tag Locking and Tag Password" for further detail.
lockaction	16bit Hex String	Lock tag Action Work setting. Require by writeoption=5 only. Refer to Section "Tag Locking and Tag Password" for further detail.
password	32bit Hex String	Tag password for the tag writing operation. Do not provide if no password is set for the tag. When writeoption is 0~4, it is the Access Password (optional). When writeoption is 5, it is the Access Password (required) When writeopting is 6, it is the Kill Password (required)

Parameter in key "tagselection" JSON object

Key	**Value Type**	**Description**
option	Integer 0~4	Tag Selection mode. 0 = Disable Tag Selection Filter Feature. 1 = Tag Selection Filter with Tag EPC. (key "addr" not required) 2 = Tag Selection Filter with data in TID Memory Bank 3 = Tag Selection Filter with data in USER Memory Bank 4 = Tag Selection Filter with data in EPC Memory Bank
addr	32bit Hex String	Define the starting address in the Tag Memory to compare with the key "data". If bit address is 0x20, it will have value "20". Please refer to RFID Tag Memory Map Layout for the memory addressing.

lenbit	Integer	Specified number of BIT to compare/match during tag filter selection. The bit comparing starts with the MSB.
data	String of Hex Byte, Max 64byte	Define the data to compare against the value in the Tag memory. The system will start to match/compare from the MSB bit for the first byte and total bit to compare is according to key "lenbit".
invert	Boolean	Optional Key. If set to True, it will select the Tag that **NOT** match the tag selection filter criteria. Default value is false.

Below are the writing request examples.

Example 1, Replace Tag's EPC without Password

In this example, once the reader accepts the command, it will return "result" = True in RPC Reply Topic.

Since no Access password is provide, and the filter is disable, the reader will select any TAG firstly detected that did not lock with password and replace the tag EPC with "EA000001".

```
{
  "r.uhfrfid.writetag": {
    "writeoption": 4,
    "writedata": "ea000001"
  }
}
```

```
{
  "deviceid": "WIRIO3_58BF25A85028",
  "pktno": 3803553077,
  "sec": 1666408719,
  "t.uhfrfid.writetag": {
    "writeoption": 4,
    "code": "0000"
  }
}
```

After the reader finished processing the command, the reader will return the write request result in the Telematic Topic. If the writing into the tag is successful, it will return "code":"0000".

Example 2, Replace EPC with Access Password

This example will cause the reader to replace the tag EPC with “EA000001” with the Tag access password set to “11112222”.

The Tag’s EPC Memory bank must be in locked condition prior to calling this writing request since the password is provided.

```
{
  "r.uhfrfid.writetag": {
    "writeoption": 4,
    "writedata": "ea000001",
    "password": "11112222"
  }
}
```

Example 3, Replace Kill/Access Password

```
{
  "r.uhfrfid.writetag": {
    "writeoption": 0,
    "writeaddr": "0",
    "writedata": "aaaabbbb11112222",
    "tagselection": {
      "option": 1,
      "lenbit": 32,
      "data": "EA000001"
    }
  }
}
```

With tag selection filter (“tagselection”) provided, “option” = 1. The total bit to check with the tag EPC, is 32 bits (“lenbit” = 32, which is 4byte). The reader will only select tag with EPC = “EA000001”. Once detected the tag, the reader will write the value provided by “writedata” (Total 4 words), into the Tag’s RESERVED Bank starting from address 0x00 (The Tag’s RESERVED Memory Bank must not locked since no password is provided)

As the Kill password memory location is at Bit address 0x00 and Access password memory location is at address bit address 0x10 (refer to section Tag Locking and Tag Password), this JSON RPC request will effectively replace the Kill password with hex value “aaaabbbb” and Access password with hex value “11112222”. The password is 32bit (4byte).

Example 4, Lock Tag’s EPC Memory Bank

This Command will Lock the Tag’s EPC Memory bank Write access. The Access Password is require and must be the same value with the value in the Tag’s RESERVED Bank Access password.

Due to the selection option, it will only write to the tag having EPC=”EA000001”.

After this command, any tag writing request to the Tag’s EPC Memory Bank, will require to provide the Access Password.

```
{
  "rpcreq": {
    "r.uhfrfid.writetag": {
      "writeoption": 5,
      "lockmask": "0020",
      "lockaction": "0020",
      "password": "11112222",
      "tagselection": {
        "option": 1,
        "lenbit": 32,
        "data": "EA000001"
      }
    }
  }
}
```

Example 5, Kill Tag

```
{
  "r.uhfrfid.writetag": {
    "writeoption": 6,
    "password": "aaaabbbb",
    "tagselection": {
      "option": 1,
      "lenbit": 32,
      "data": "EA000001"
    }
  }
}
```

This RPC Request will select the tag having EPC = “EA000001” and send the Kill tag command with the Kill Password = “aaaabbbb” to the tag.

Once the kill tag command executed successfully, the tag will no longer usable. It will not respond to any reader request anymore.

Tag Reading

The RFID Tag reading feature includes,

- Reading one tag at a time.
- Optional tag selection filter according to data in any memory bank.
- Reading on tag memory bank RESERVED, EPC, TID and USER with user definable writing data size and address location in the tag memory. Maximum number of bits on each read is 96 words (1536bits)
- Tag reading data must be in multiple of 1 word (16 bit/2 bytes).

Parameter require for the Tag Reading request will be placed under key value "r.uhfrfid.readtag" JSON object. Users can request Tag Reading while the reader is idling or while the reader is on Tag Inventory/Reading mode.

If the parameter of Tag Reading request command provided fulfils the requirement and accepted for processing, it will respond with result = TRUE in RPC Reply Topic. After that, the request will be queue for processing. Once the request is processed, the processed result will be returned in Telematics Topic under key "t.uhfrfid.readtag".

If the system is currently busy or the request command provided does not fulfill the parameter requirement, it will respond with result = FALSE.

For more information on the Tag Reading telematics result, please refer to "t.uhfrfid.readtag" under "Telematics Data" Section.

Key	Value Type	Description
r.uhfrfid.readtag	JSON Object	All the parameters required will be placed under this JSON object

Parameter in key "r.uhfrfid.writetag" JSON object

Key	Value Type	Description
readoption	Integer 0~3	Tag Writing process available option are, 0 = Read to RESERVED memory bank 1 = Read to EPC memory bank 2 = Read TID memory bank 3 = Read USER memory bank
timeout	Integer, 16bit	Optional. Tag writing time out in millisecond. Default value is 1000ms.
tagselection	JSON object	Optional. JSON object that provides the tag selection/filter criteria when writing the tag. If not provided, it will write to all the surrounding detected tag. For further detail, please refer to the Parameter in key "tagselection" JSON object at "Tag Writing -> Parameter in Key tagselection".
readaddr	32bit Hex String	Set the start address (16bit addressing) of the tag memory to read.

		E.g. If Bit Address of the Tag Memory is 0x20, the readaddr will be set to "2". Please refer to RFID Tag Memory Map Layout for the bit addressing.
wordcount	Integer 1~96	Number of words(16bits) to read. Maximum 96words.
password	32bit Hex String	Tag's Access password during tag reading operation for locked tag. Do not provide if no password is set for the tag.

Example 1: Reading any surrounding tag

```
{
  "r.uhfrfid.readtag": {
    "readoption": 1,
    "readaddr": "0",
    "wordcount": 5
  }
}
```

The device will start to read any first detected tag. It will read the EPC Memory Bank, starting from Word Address 0x0. It will read five words (10bytes) in total.

When the device receives the request and the request provided fulfils the requirement, the device will send out ack respond JSON packet in the RPC response topic.

```
{
  "deviceid": "WIRIO3_58BF25A85028",
  "rssi": -64,
  "pktno": 1428894619,
  "r.uhfrfid.result": true
}
```

If the reading is successful, it will return the reading result in the Telematic Topic with code="0000" together with the tag data requested.

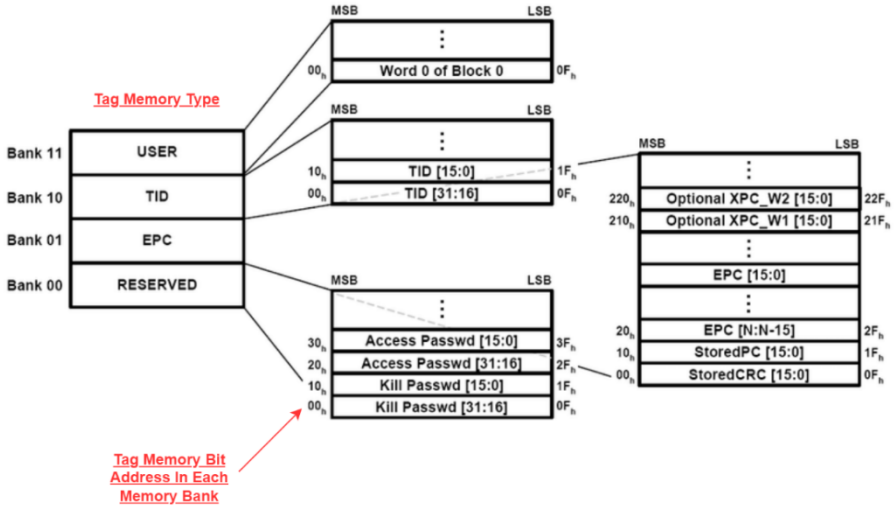
```
{
  "deviceid": "WIRIO3_58BF25A85028",
  "rssi": -69,
  "pktno": 1428894620,
  "sec": 1666753422,
  "t.uhfrfid.readtag": {
    "readoption": 1,
    "code": "0000",
    "tagdata": "191E1420EA0000017084"
  }
}
```

Example 2: Reading Tag with Selection Filter

This command will only read tag with Tag EPC= "EA000001". It will read the tag's User Memory Bank starting from Word Address 0x0, and will total five words (10bytes)

```
{
  "r.uhfrfid.readtag": {
    "readoption": 3,
    "readaddr": "0",
    "wordcount": 5,
    "tagselection": {
      "option": 1,
      "lenbit": 32,
      "data": "EA000001"
    }
  }
}
```

RFID Tag Memory Map Layout



The Tag contains 4 types of Memory bank, Reserved, EPC, TID and USER. The addressing for the memory in each bank is according to BIT addressing.

E.g., The EPC data for the tag is located at Bank 01 (EPC) starting from address 0x20. Please take note that Address 0x20 will be BIT15 of the EPC data and Address 0x2F will be BIT0 of the EPC data.

The default addressing method shown in the memory map above is based on bit addressing. Each memory bank has its own bit address and all of them start from bit address 0x00.

As for Word addressing, each Word has 16bit. Example Word addressing conversion from bit addressing are as follows,

Bit address 0x00 = Word address 0x00

Bit address 0x10 = Word address 0x01

Bit address 0x20 = Word address 0x02

When reading or writing into the Tag memory, it will write in multiple of 16bit only.

Tag Selection/Singulation

The reader allows the user to configure the Tag Selection/Singulation Filter according to the parameter provided during the Tag Inventory/Reading process and Tag writing process (with RPC Request).

The tag selection can be based on criteria below,

- Compare the filter byte provided with any memory location in the 4-memory bank or with the Tag's EPC.
- Compare any number of bits with the filter byte provided.
- Invert the selection to select the Tag that does NOT fulfill the criteria provided.

Example

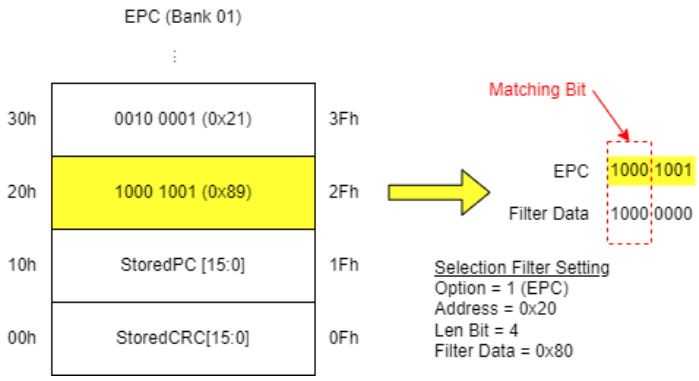
The selection criteria requirement is as below,

- Compare the filter byte provided with EPC Memory Bank starting from bit address = 0x20.
- The total number of bits to compare is 4 bits.
- The filter byte provided is 0x80.

The attribute setting for the reader during Tag Inventory/Reading are as below,

Attribute setting for Tag Inventory/Reading	*RPC Request under "tagselection" during Tag Writing Request*	*Value*	*Value Type*
s.uhfrfid.selfilteroption	option	4	Integer
s.uhfrfid.selfilteraddr	addr	"20"	Hex String
.uhfrfid.selfilterdata	data	"80"	Hex String
s.uhfrfid.selfilterlenbit	lenbit	4	Integer
s.ufhrfid.selfilterinvert	invert	false	Boolean

With the setting above, the device will only compare the first 4 bits of the Filter data with the EPC Memory bank starting from bit address 0x20 to 0x23.



Tag Locking and Tag Password

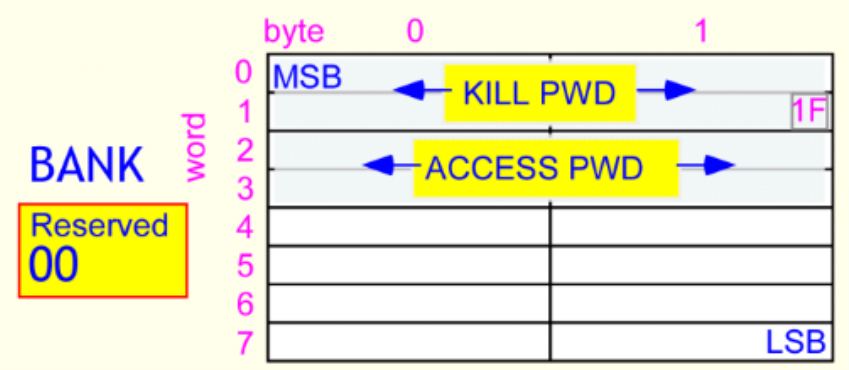
Locking the EPC Gen2 tag will require setting the access password to none zero and set the locking state of each memory bank (Perma-Lock/Perma-Unlock Bit and Write Lock Bit).

Gen2 Tag Passwords

An EPC Gen2 tag has two separate passwords, an Access Password and a Kill Password, each are 32 bits and are stored in the reserved bank (bank 00) of the tag memory. Both passwords can be separately lock from future reading and writing.

The reader is requiring using Access Password when the reader tries to read/write to the tag that having the Access Password already set in the RESERVER Bank.

To Kill/Disable the tag, user require to provide the Kill Password. The “Kill” Tag is not accessible from any RFID reader anymore.



Lock Mask and Lock Action

	First Byte								Second Byte							
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory Zone	Reserved Set to 0						Kill Pwd		Access Pwd		EPC Bank		TID Bank		User Bank	
Mask							Set	Set	Set	Set	Set	Set	Set	Set	Set	Set
Action							R/W	Perm	R/W	Perm	W	Perm	W	Perm	W	Perm

1. R/W = Access Password Read and Write Lock
2. W = Access Password Write Lock
3. Perm = Perma-Lock (1)/Perma-Unlock (0)

When the Bit in Mask Word is set to 1, the system will use the correspondent bit’s function in the Action Word.

E.g.,

If Mask Word is 0x0020 and Action Word is 0x0020, EPC Bank will be Write Locked.

Tag Locking State Chart

Below is the possible locking state of the tag by setting the Perma bit and W(R/W) bit,

	W (R/W) = 0	**W (R/W) = 1**
Perma = 0, (Perma-unlocked)	You can read and write the Tag memory, and you can change the protection status. Tag data is unsecure, and the user is allowing to change (Read and write) the data any time.	You can read the memory zone (If it is R/W Lock, reading also requires access passwords), but to write it, you must provide the access password. This means that the memory zone is password write protected. You can also change the protection status. This is the preferred way to reversibly unlock the EPC memory.
Perma = 1, (Perma-locked)	You can read and write the EPC memory. But you cannot change the protection status anymore. Tag data is unsecure, and the user is allowing to change (Read and write) the data any time.	You can read the memory zone, but you can never write it again. You will never be able to change the protection status. This is the preferred way to permanently lock the memory zone.

The Perma bit can only set **ONCE** only. Once it is set, it cannot be clear.

If Perma bit is **NOT** set, the R or R/W bit can be set or clear. But if the Perma bit is set, the R or R/W bit will be permanently lock and not allowed to change.

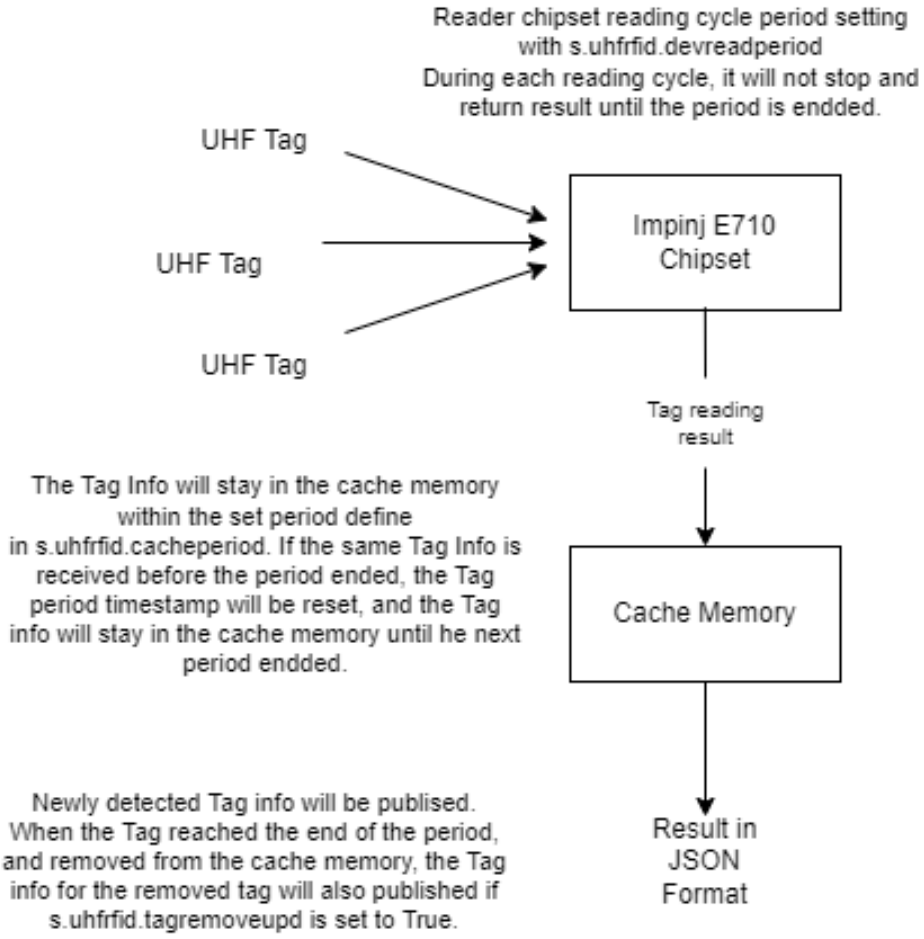
Tag Locking process.

The steps to lock the tag are as below,

1. Write a 32bit non-zero Access Password and Kill Password.
2. Program the data into the memory bank that needs to lock.
3. Lock the memory bank and the Access and Kill password memory to prevent reading of the access password and kill password.

Please note that only reserved memory bank (Access and Kill Passwords) can be both READ and WRITE locked - all others (EPC, TID, and User) can write-locked only. Typically, the Tag Identification (TID) memory bank perma-locked (set to 1) at the factory during tag manufacturing.

Tag Reading Process



During the Tag reading process, the Tag info detected by the Impinj Reader Chipset will only be send out the MCU Cache memory when reached the end of the s.uhfrfid.devreadperiod period.

The Tag information will stay in the cache for the period of s.uhfrfid.cacheperiod. If the same Tag detected before reaching the end of the period, the tag time stamp will reset. The tag will only remove from the Tag Cache Memory when it reaches the end of the Tag Cache period.

When adding the Tag into the Tag Cache Memory for the first time, the tag information will send out in JSON format to the backend system (through MQTT server or serial port).

When the tag is removed from the cache memory, if s.uhfrfid.tagremoveupd is set to true, the tag information will be sent out in JSON format to the backend system too.

When the reading stops, all the Tag information in the cache memory will delete silently.

TAG Reading/Writing Return Code

Error Code	Description
0000	Operation Success
0100	Write Data Length Error
0105	Parameter Error
0400	No Tag Detected
040A	Tag Writing Error
0420	GEN2 Other Error
0423	Tag Memory Overrun, Bad PC
0424	Memory Locked Unable to Write
042B	Insufficient Power
042F	Non-Specific Error
0430	Unknow Error
0504	Reader Over Temperature
0505	Standing Wave Ratio/Reflection too large
f000	Command Sent Error