Macrotech Lock Controllers

Use cases and Message Payloads

Last updated on: 12 Nov 2022

Table of Contents

[**Introduction**](#_39p6cl3ktedd) **4**

[**Messages - Packet Structures and JSON Payload**](#_fxgs24v28ydn) **5**

[**Message Data Types**](#_dt4z0jwzctcb) **6**

[**Device Id - Mapping**](#_pf0dq7xwksic) **16**

[**Use cases**](#_3phfkf2mvbwx) **16**

[Get Status - Individual Lock](#_sujl27bi7b86) 17

[JSON Payload - Cloud IoT Server to IoT Gateway](#_l379482sb34g) 17

[PDU Structure - IoT Gateway to a specific Lock controller node](#_wb3evhharqgh) 17

[Response for Get Status - Individual Lock](#_u0n591h1njzo) 18

[PDU Structure - Lock controller node to IoT Gateway](#_ck4xwk76eej) 18

[JSON Payload - IoT Gateway to Cloud IoT Server](#_4a8d5dudq0g) 19

[Get Status from All Lock Controllers](#_eqs968szh6ap) 20

[PDU Structure - IoT Gateway to all Lock controller nodes](#_3e7qenomc04r) 21

[Response for Get Status From All Lock Controllers](#_mupkyr6t4lyf) 21

[PDU Structure - Lock controller node to IoT Gateway](#_f3vsnwaestvy) 21

[JSON Payload - IoT Gateway to Cloud IoT Server](#_andgshze8veb) 21

[Unlock Door - Normal Operation](#_3i80krwftt5d) 21

[JSON Payload - Cloud IoT Server to IoT Gateway](#_7ghm4j11fp8b) 21

[PDU Structure - IoT Gateway to a specific Lock controller node](#_zi4ev4dfgu8n) 22

[Response for Unlock Door - Normal Operation](#_6vuicsvk4hjn) 22

[PDU Structure - Lock controller node to IoT Gateway](#_w4q3pdlfx4kg) 22

[JSON Payload - IoT Gateway to Cloud IoT Server](#_r7e7tsloheh5) 22

[Unlock Door - Emergency Operation](#_80gu1mt8ta3j) 23

[JSON Payload - Cloud IoT Server to IoT Gateway](#_hn4cwadtd4t4) 23

[PDU Structure - IoT Gateway to a specific Lock controller node](#_3jroj0g28r7v) 23

[Response for Unlock Door - Emergency Operation](#_azac3sx39pca) 24

[PDU Structure - Lock controller node to IoT Gateway](#_jwyzc5fm4o8g) 24

[JSON Payload - IoT Gateway to Cloud IoT Server](#_dt0hou47x9gc) 24

[Enable Autolock](#_xm4s41bietau) 25

[JSON Payload - Cloud IoT Server to IoT Gateway](#_nopt8m2k4ifz) 25

[PDU Structure - IoT Gateway to a specific Lock controller node](#_l9pnx7j5ujc2) 25

[Response for Enable Autolock](#_6oaa5rixui0v) 26

[PDU Structure - Lock controller node to IoT Gateway](#_tdepif4vrhf) 26

[JSON Payload - IoT Gateway to Cloud IoT Server](#_dicwpin2rvbx) 26

[Disable Autolock](#_wix384xrzepr) 27

[JSON Payload - Cloud IoT Server to IoT Gateway](#_3o8zp6o5ykt8) 27

[PDU Structure - IoT Gateway to a specific Lock controller node](#_v7vyhtuu5283) 27

[Response for Disable Autolock](#_i3j1obkajkp4) 27

[PDU Structure - Lock controller node to IoT Gateway](#_om67revsr1sk) 27

[JSON Payload - IoT Gateway to Cloud IoT Server](#_xl74zrhvcsh0) 28

[Disable Autolock On All Lock Controllers - Emergency Operation](#_n06o5nm5tsk) 28

[PDU Structure - IoT Gateway to all Lock controller nodes](#_lsf3f2kwep0r) 29

[Response for Disable Autolock On All Lock Controllers - Emergency Operation](#_qsy5iwy3ala) 29

[PDU Structure - Lock controller node to IoT Gateway](#_dy8dpeuhak2o) 29

[JSON Payload - IoT Gateway to Cloud IoT Server](#_cvxiv3do0f13) 29

[Set Master Data](#_iwpezw3b7g4r) 29

[JSON Payload - Cloud IoT Server to IoT Gateway](#_w6ck2smw40ro) 30

[PDU Structure - IoT Gateway to a specific Lock controller node](#_sgjjw4t0756x) 30

[Response for Set Master Data](#_utyoffdw5iqd) 31

[PDU Structure - Lock controller node to IoT Gateway](#_hopx8h3lug7m) 31

[JSON Payload - IoT Gateway to Cloud IoT Server](#_uujpb33m4ee) 31

[Set Admin Key](#_bmknv9qwam83) 32

[JSON Payload - Cloud IoT Server to IoT Gateway](#_xrs763hlp9zk) 32

[PDU Structure - IoT Gateway to a specific Lock controller node](#_druck2dtscnq) 32

[Response for Set Admin Key](#_k5lrt7efqv81) 33

[PDU Structure - Lock controller node to IoT Gateway](#_6oswo3hg5sr7) 33

[JSON Payload - IoT Gateway to Cloud IoT Server](#_jx05iuegmids) 33

[Set Admin Key - All Lock Controllers](#_edssf191neid) 34

[JSON Payload - Cloud IoT Server to IoT Gateway](#_2oz1yof2akde) 34

[PDU Structure - IoT Gateway to all Lock controller nodes](#_me8v6rcub86k) 35

[Response for Set Admin Key on All Lock Controllers](#_8t7fgl43jyjq) 35

[PDU Structure - Lock controller node to IoT Gateway](#_av8lawpx3qaz) 35

[JSON Payload - IoT Gateway to Cloud IoT Server](#_6gepg84l8q1t) 35

[Set Staff Key](#_1y7rji9057o2) 35

[JSON Payload - Cloud IoT Server to IoT Gateway](#_iy8i5pgncr45) 35

[PDU Structure - IoT Gateway to a specific Lock controller node](#_vc1vtctetd8p) 36

[Response for Set Staff Data](#_mskpcuibwalf) 36

[PDU Structure - Lock controller node to IoT Gateway](#_50pua55d5llq) 36

[JSON Payload - IoT Gateway to Cloud IoT Server](#_p8lejyypncq5) 36

[Set Guest Key](#_dlb92md1sout) 37

[JSON Payload - Cloud IoT Server to IoT Gateway](#_581jg8m6bmxx) 37

[PDU Structure - IoT Gateway to a specific Lock controller node](#_jx13qnewspwz) 37

[Response for Set Guest Key](#_wp8mpzv5v87y) 38

[PDU Structure - Lock controller node to IoT Gateway](#_z8pdo4smrlkg) 38

[JSON Payload - IoT Gateway to Cloud IoT Server](#_iq2kwqdc0nos) 38

[Delete RFID Data](#_xaufpo1mq72v) 39

[JSON Payload - Cloud IoT Server to IoT Gateway](#_uf24zt3lbepd) 39

[PDU Structure - IoT Gateway to a specific Lock controller node](#_xaihlpcv0r5g) 39

[Response for Delete RFID Data](#_p9jk8orpkysg) 40

[PDU Structure - Lock controller node to IoT Gateway](#_vkw62vkvr9i3) 40

[JSON Payload - IoT Gateway to Cloud IoT Server](#_ahse7h3us8dx) 40

[Delete RFID Card Data from All Lock Controllers](#_1isvksna7r4u) 41

[JSON Payload - Cloud IoT Server to IoT Gateway](#_c1ww0f8au22v) 41

[Heartbeat - Periodic Telemetry Message](#_l8vzn55yp2hi) 45

[PDU Structure - Lock controller node to IoT Gateway](#_546e1y1vcbfy) 45

[JSON Payload - IoT Gateway to Cloud IoT Server](#_o2ttv01k3imv) 46

[Status Update Message](#_x4z1spnzm3kz) 46

[PDU Structure - Lock controller node to IoT Gateway](#_y9uqn4m7gsvq) 47

[JSON Payload - IoT Gateway to Cloud IoT Server](#_uheawern1cbw) 47

[Alarm - Low-Battery](#_ti1nqzw4vh44) 47

[PDU Structure - Lock controller node to IoT Gateway](#_sj3spqaw72of) 47

[JSON Payload - IoT Gateway to Cloud IoT Server](#_5pvgu6k86gob) 47

[Alarm - Deadbolt Alert](#_g68afzkxkwnz) 48

[PDU Structure - Lock controller node to IoT Gateway](#_gbk2pod8rzoh) 48

[JSON Payload - IoT Gateway to Cloud IoT Server](#_bupgge1v2p9e) 48

[Alarm - Latch Alert](#_7iikg1bo1rh) 49

[PDU Structure - Lock controller node to IoT Gateway](#_u95jkm2gjovt) 49

[JSON Payload - IoT Gateway to Cloud IoT Server](#_2i10fd19n9m5) 49

[Alarm - Unlock by RFID Card Alert](#_o51hxj3vmdu4) 49

[PDU Structure - Lock controller node to IoT Gateway](#_qa0nwyj1czps) 49

[JSON Payload - IoT Gateway to Cloud IoT Server](#_9pgfhlgkexak) 50

[**Appendix A - Wirepas Node Address and Device Id mapping CSV file**](#_tnxjvkvd5h8b) **50**

[**Appendix B - Persistent Storage in Lock Controller for storing Master, Staff and Guest Data**](#_4h94ukq0pk3e) **51**

# Introduction

This document lists various use cases pertaining to managing the RF network of Macrotech’s Lock controllers along with the corresponding message exchanging payload structures between:

* Cloud IoT Platform (Macrotech’s CloudExt) <-> IoT Gateway
* IoT Gateway <-> Wirepas Sink node <-> Lock controller nodes

# Messages - Packet Structures and JSON Payload

Between Cloud IoT Platform and IoT Gateway, the messages are exchanged in JSON payload using MQTT protocol.

Between IoT Gateway and Lock controller nodes, the messages are exchanged as encoded packets of bytes via Wirepas Sink node. The structure of this encoded packets is given below:

| **Byte:Length** | **Field Name** | **Description** |
| --- | --- | --- |
| 0:1 | Beginning Marker | Beginning of Packet Marker. Should be 0xAA. |
| 1:1 | Message Type | First 2 Most Significant Bits (MSBs) specify destination address type:  01 - Unicast  10 - Multicast  11 - Broadcast  Next 3 bits specify the operation type:  000 - Get  001 - Set  010 - Notification  011 - Acknowledgement  Last 3 bits specify the type of data:  000 - Attribute  001 - State  010 - Telemetry  011 - Alarm |
| 2:4 or 1 or 0 | Source/Destination Address | * For messages sent from the individual node, the source address should be 4 bytes containing Wirepas node address of that individual node. * For messages sent from the sink node to an individual node, the destination address should be 4 bytes Wirepas node address of the target node. * For multicast messages sent from the sink node, this field is 1 byte containing the group number. * For broadcast messages sent from the sink node, this field should be absent (0 bytes). |
| 6 or 3 or 2:4 | Request ID | 4 bytes signed int used as the request identifier. The nodes should process the request id field in the following way:   * When a Get or Set type of packet is received, this request id should be sent back when sending the response for Get operation (Notification) or Set operation (Acknowledgement). This is very important as the gateway uses the request id to map the corresponding response it sends to Cloud IoT Server. * While sending Telemetry or Alarm messages, the request id should be -1. |
| 10 or 7 or 6:1 (optional) | Key and Length | First 4 Most Significant Bits (MSBs) specify the key of the data element.  With 4 bits, there can be 15 keys for each data type (Attribute, State, Telemetry and Alarm). Note that only 0x1 to 0xF are used (0x0 is not used).  Last 4 bits specify the length of the data element. With 4 bits, the data length can vary upto a maximum of 15 bytes. If no value is specified, then 0x0 is used. For example, in Get messages, only the key field is passed without any value. |
| 11 or 8 or 7:x | Value | If the data element’s length in the previous byte is non-zero, then the corresponding value is captured in N number of bytes, where N = Length as specified in the previous byte. |
| x:1 | End Marker | End of Packet Marker. Should be 0x00 (for this reason, 0x0 is not used for the key).  The packet may contain more than 1 data element and the end marker is used to mark the end of Key-Length-Value tuples. |

# Message Data Types

The following tables provide the list of supported lock controller data types (Attribute, State, Alarm and Telemetry) in Lock controller nodes (Wirepas RF mesh) and Cloud IoT Server (CloudExt IoT Platform).

| **Attribute** | | | |
| --- | --- | --- | --- |
| Device ID | Wirepas specific data type | Key | **0x10** |
| Data type | **Byte** |
| Size | **6** |
| Type | **Read-only** |
| IoT Server specific data type | ***macaddr*** - String in the form of xx-xx-xx-xx-xx-xx | |
| This field provides mapping of Device to it's hardware ID. Useful to map server side name to device side ID. Last 3 bytes of this ID are used by Wirepas as a Node address. | | |
| Firmware Version | Wirepas specific data type | Key | **0x20** |
| Data type | **Byte** |
| Size | **2** |
| Type | **Read-only** |
| IoT Server specific data type | ***verstr*** - String in the form {major}.{minor}.{maint}.  For example: 1.4.21, 1.0.00 | |
| Wirepas side:  1st byte contains 4 bits major version no. and 4 bits minor version number, and 2nd byte contains maintenance version number. Example 0x10 0x01 means version 1.0.1 | | |
| Location | Wirepas specific data type | Key | **0x30** |
| Data type | **String** |
| Size | **15** |
| Type | **Read Write** |
| IoT Server specific data type | String.  For example: “F1-101-Lock” | |
| This field contains free form text upto 15 characters long. It is used for uniquely naming the device to identify it’s geographical location. | | |
| Telemetry Sampling Interval | Wirepas specific data type | Key | **0x40** |
| Data type | **Short Integer** |
| Size | **2** |
| Type | **Read Write** |
| IoT Server specific data type | Integer | |
| Represents the telemetry sampling interval (periodicity) in seconds. For disabling telemetry, the value should be set 0. | | |

| **State** | | | |
| --- | --- | --- | --- |
| lock | Wirepas specific data type | Key | **0x10** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read Write** |
| IoT Server specific data type | Boolean (true/false) | |
| Used for activating or deactivating the lock when the deadbolt is disengaged. If 'true', activates the lock. For deactivating the lock, the value should be 'false'.   * For activating the lock, the byte value should be 0x01, which is equivalent to Boolean true. * For deactivating the lock, the byte value should be 0x00, which is equivalent to Boolean false. * As of now, only deactivation of lock is supported for “Set” operation. | | |
| emergency-lock | Wirepas specific data type | Key | **0x20** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read Write** |
| IoT Server specific data type | Boolean (true/false) | |
| Used for activating or deactivating the lock in emergency situations (when the deadbolt is engaged). If 'true', activates the lock. For deactivating the lock, the value should be 'false'.   * For activating the lock, the byte value should be 0x01, which is equivalent to Boolean true. * For deactivating the lock, the byte value should be 0x00, which is equivalent to Boolean false. * As of now, only deactivation of lock is supported for “Set” operation. | | |
| status | Wirepas specific data type | Key | **0x30** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | ***status*** - String (“online” or “offline”) | |
| Represents the current latch status.   * If closed, the value of the byte is going to be 0x01, which is equivalent to “online”. * If opened, the value of the byte is going to be 0x00, which is equivalent to “offline”. | | |
| battery | Wirepas specific data type | Key | **0x40** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | ***char*** - Integer with a value in the range 0-100 | |
| Represents the current charge level of the battery in percentage. The value ranges from 0 to 100. | | |
| deadbolt | Wirepas specific data type | Key | **0x50** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| Used for providing the current status of the deadbolt.   * If the deadbolt is engaged, the value of the byte is going to be 0x01, which is equivalent to Boolean true * If the deadbolt is disengaged, the value of the byte is going to be 0x00, which is equivalent to Boolean false | | |
| latch | Wirepas specific data type | Key | **0x60** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| Used for providing the current status of the latch..   * If the latch is closed, the value of the byte is going to be 0x01, which is equivalent to Boolean true * If the latch is opened, the value of the byte is going to be 0x00, which is equivalent to Boolean false | | |
| autolock | Wirepas specific data type | Key | **0x70** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read Write** |
| IoT Server specific data type | Boolean (true/false) | |
| Used for enabling or disabling the auto-lock. If 'true' enables the auto-lock, so that users have to use NFC or web interface to open the lock. If ‘false’ disables the auto-lock so that door always remains unlocked. | | |
| master\_data | Wirepas specific data type | Key | **0x80** |
| Data type | **master\_data** |
| Size | **11** |
| Type | **Read Write** |
| IoT Server specific data type | master\_data JSON object containing the following keys and values:  {  "macrotech\_id" : <unsigned int>,  "hotel\_id" : <unsigned short int>,  "lock\_type" : "guest\_room" or "common\_door",  "floor" : 0 to 255,  "room\_number" : <unsigned short>,  “autolock” : true/false  } | |
| Used for setting master data | | |
| admin\_key | Wirepas specific data type | Key | **0x90** |
| Data type | **Unsigned Integer** |
| Size | **4** |
| Type | **Read Write** |
| IoT Server specific data type | admin\_key JSON object containing the following keys and values:  {  "key" : <unsigned int>  } | |
| Used for setting the admin key. | | |
| staff\_key | Wirepas specific data type | Key | **0xA0** |
| Data type | **Unsigned Integer** |
| Size | **4** |
| Type | **Read Write** |
| IoT Server specific data type | staff\_key JSON object containing the following keys and values:  {  "key" : <unsigned int>  } | |
| Used for setting the staff key. | | |
| guest\_key | Wirepas specific data type | Key | **0xB0** |
| Data type | **Unsigned Integer** |
| Size | **4** |
| Type | **Read Write** |
| IoT Server specific data type | guest\_key JSON object containing the following keys and values:  {  "key" : <unsigned int>  } | |
| Used for setting the guest key. | | |
| delete\_rfid\_data | Wirepas specific data type | Key | **0xC0** |
| Data type | **card\_data** |
| Size | **5** |
| Type | **Read Write** |
| IoT Server specific data type | {  “card\_id” : <unsigned int>,  “card\_type” : “master” | “staff” | “guest”  }  **card\_id** - Unsigned Int. If set to zero, ignore this field. If non-zero, then delete this specific card, if present (in this case, card\_type field should be ignored).  **card\_type** - Byte. Can be 0x0 (ignore), 0x1 (master), 0x2 (staff), 0x3 (guest) or 0xFF (all). If card\_id is not zero. This field should be used only if card\_id is set to 0. In case of card\_type, all card ids belonging to the given card\_type should be deleted. Note that, if card\_id is set to non-zero, then this field shall be set to 0x0 (ignore). If card\_type is set to 0xFF (all), then all card ids should be deleted. | |
| Used for deleting/erasing RFID card data stored in the lock controller. | | |
| firmware\_version | Wirepas specific data type | Key | **0xD0** |
| Data type | **Byte** |
| Size | **2** |
| Type | **Read-only** |
| IoT Server specific data type | ***verstr*** - String in the form {major}.{minor}.{maint}.  For example: 1.4.21, 1.0.00 | |
| Used for providing the information about the current firmware version.  Wirepas side:  1st byte contains 4 bits major version no. and 4 bits minor version number, and 2nd byte contains maintenance version number. Example 0x10 0x01 means version 1.0.1 | | |
| date\_time | Wirepas specific data type | Key | **0xE0** |
| Data type | **Unsigned Integer** |
| Size | **4** |
| Type | **Read Write** |
| IoT Server specific data type | ***String*** - String in the form YYYY-MM-YY, hh:mm:ss representing: Year as YYYY (for example: 2022)  Month as MM ranging from 01 to 12  Day as DD ranging from 01 to 31  Hours as hh ranging from 00 to 23 (24-hr format)  Minutes as mm ranging from 00 to 59  Seconds as ss ranging from 00 to 59 | |
| Used for setting/retrieving the current date and time on the lock controller.  Wirepas side:  4 byte unsigned integer representing the date and time in epoch seconds (seconds elapsed since 00:00:00 UTC on 1 January 1970).  **IMPORTANT NOTE:**  Since lock controller nodes do not have the timezone setting feature, the gateway sends the epoch seconds (UTC) as is without adjusting the seconds to the local timezone. Likewise, when it receives the epoch seconds from the lock controller nodes, it simply converts the epoch seconds to YYYY-MM-DD… form without adjusting for the local timezone. | | |

| **Alarm** | | | |
| --- | --- | --- | --- |
| low-battery | Wirepas specific data type | Key | **0x10** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true) | |
| This alarm is triggered when the battery level drops below 10%. When the alarm is generated, it’s value is always set to 0x01, which is equivalent to Boolean true | | |
| deadbolt-alert | Wirepas specific data type | Key | **0x20** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| This alarm is triggered when the deadbolt is engaged or disengaged. When the deadbolt is engaged, the corresponding value is set to 0x01 (Boolean true) and when the deadbolt is disengaged, the value is set to 0x00 (Boolean false) | | |
| latch-alert | Wirepas specific data type | Key | **0x30** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| This alarm is triggered when the latch is opened and closed. When the latch is closed, the corresponding value is set to 0x01 (Boolean true) and when the latch is opened, the value is set to 0x00 (Boolean false). | | |
| unlock-by-rfid-alert | Wirepas specific data type | Key | **0x40** |
| Data type | **unlock\_alert\_data** |
| Size | **6** |
| Type | **Read-only** |
| IoT Server specific data type | {  “card\_id” : <unsigned int>,  “card\_type” : “admin” | “staff” | “guest”,  “status” : “success” | “failed”  } | |
| This alarm is triggered when an attempt is made to unlock the door using an RFID card. The alarm contains the following data:   * Card id - <unsigned int> - First 4 bytes * Card type - 1 byte containing 0x01 (admin card) or 0x02 (staff card) or 0x03 (guest card) * Status - 1 byte containing 0x01 (success) or 0x00 (failed) | | |

| **Telemetry** | | | |
| --- | --- | --- | --- |
| status | Wirepas specific data type | Key | **0x10** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | ***status*** - String (“online” or “offline”) | |
| Represents the current lock status, which represents the latch status.   * If the latch is closed, the value of the byte is going to be 0x01, which is equivalent to “online”. * If the latch is opened, the value of the byte is going to be 0x00, which is equivalent to “offline”. | | |
| battery | Wirepas specific data type | Key | **0x20** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | ***char*** - Integer with a value in the range 0-100 | |
| * Represents the current charge level of the battery in percentage. The value ranges from 0 to 100. | | |
| deadbolt | Wirepas specific data type | Key | **0x30** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| Represents the current deadbolt status.   * If the deadbolt is engaged, the value of the byte is going to be 0x01, which is equivalent to Boolean true. * If the deadbolt is disengaged, the value of the byte is going to be 0x00, which is equivalent to Boolean false. | | |
| latch | Wirepas specific data type | Key | **0x40** |
| Data type | **Byte** |
| Size | **1** |
| Type | **Read-only** |
| IoT Server specific data type | Boolean (true/false) | |
| Represents the current latch status.   * If the latch is closed, the value of the byte is going to be 0x01, which is equivalent to Boolean true. * If the latch is opened, the value of the byte is going to be 0x00, which is equivalent to Boolean false. | | |
| autolock | Wirepas specific data type | Key | 0x50 |
| Data type | Byte |
| Size | 1 |
| Type | Read-only |
| IoT Server specific data type | Boolean (true/false) | |
| Represents the current auto-lock status.   * If the auto-lock is enabled, the value of the byte is going to be 0x01, which is equivalent to Boolean true. * If the auto-lock is disabled, the value of the byte is going to be 0x00, which is equivalent to Boolean false. | | |

# Device Id - Mapping

Each lock controller node is mapped with a unique identifier. In Wirepas RF mesh network, the lock controller node uses *Wirepas node address*, which is a 6 byte integer value.

In CloudExt IoT Platform, each controller node is identified with a human-readable string instead of *Wirepas node address*.

The IoT Gateway performs the device id translation (Wirepas node address to CloudExt human-readable string and vice-versa) while exchanging the messages between Wirepas RF mesh network and Cloud IoT Platform.

So, it is important to provide the mapping file (CSV) containing Wirepas node address mapped to CloudExt human-readable string before deploying/starting IoT Gateway.

See [Appendix A](#_4zqfbezibtzq) for an example mapping CSV file.

# Use cases

**Note:**

* Device Id mentioned in MQTT topics represents the human-readable string used in CloudExt IoT Platform to represent a lock controller node.
* Node Address mentioned in PDU Structure represents the wirepas node address of the corresponding lock controller node in Wirepas RF mesh network.

## Get Status - Individual Lock

Cloud IoT Server initiates this operation by sending a command to get the status of a specific lock in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**getstatus**”, “Seq” : <integer> } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the status should be fetched from that lock.  The Seq field contains an integer that should be returned back as is in the response payload. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Get-State Endpoint (EP), which is **0x01**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x30 | Key = status, Length = 0 |
| 6:1 | 0x40 | Key = battery, Length = 0 |
| 7:1 | 0x50 | Key = deadbolt, Length = 0 |
| 8:1 | 0x60 | Key = latch, Length = 0 |
| 9:1 | 0x80 | Key = master\_data, Length = 0 |
| 10:1 | 0x90 | Key = admin\_key, Length = 0 |
| 11:1 | 0xA0 | Key = staff\_key, Length = 0 |
| 12:1 | 0xB0 | Key = guest\_key, Length = 0 |
| 13:1 | 0xD0 | Key = firmware\_version, Length = 0 |
| 14:1 | 0xE0 | Key = date\_time, Length = 0 |
| 15:1 | 0x00 | End of Packet Marker |

## Response for Get Status - Individual Lock

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Notification-State Endpoint (EP) for the destination, which is **0x11**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the *Get Status* request message. See sections [Get Status - All Locks](#_j06ch9v8pba4) and [Get Status - Individual Lock](#_rjwp9reavl9e) for additional information. |
| 5:1 | 0x31 | Key = status, Length = 1 |
| 6:1 | 0x0 or 0x1 | status. 0x0 means offline and 0x1 means online. |
| 7:1 | 0x41 | Key = battery, Length = 1 |
| 8:1 | 0x0 to 0x64 | Battery level in percentage. 0 (0x) to 100 (0x64). |
| 9:1 | 0x51 | Key = deadbolt, Length = 1 |
| 10:1 | 0x0 or 0x1 | Deadbolt status. 0x0 means disengaged and 0x1 means engaged. |
| 11:1 | 0x61 | Key = latch, Length = 1 |
| 12:1 | 0x0 or 0x1 | Latch status. 0x0 means opened and 0x1 means closed. |
| 13:1 | 0x8B | Key = master\_data, Length = 11 |
| 14:4 |  | 4 bytes. Unsigned Integer. Represents Macrotech ID stored. |
| 18:2 |  | 2 bytes. Unsigned Short Integer. Represents Hotel ID stored in little-endian form. |
| 20:1 |  | 1 byte. Represents the lock type.  0x01 means the lock is associated with a guest room.  0x02 means the lock is associated with a common door. |
| 21:1 |  | 1 byte. Represents the floor from 0 to 255 (0xFF). |
| 22:2 |  | 2 bytes. Unsigned Short Integer. Represents room number (if lock type is guest room) or common door (if lock type is common door) stored in little-endian form. |
| 24:1 |  | 1 byte. Represents the autolock status. 0x0 means disabled and 0x1 means enabled |
| 25:1 | 0x94 | Key = admin\_key, Length = 4 |
| 26:4 |  | 4 bytes. Unsigned Integer. Represents Admin Key stored in little-endian form. |
| 30:1 | 0xA4 | Key = staff\_key, Length = 4 |
| 31:4 |  | 4 bytes. Unsigned Integer. Represents Staff Key stored in little-endian form. |
| 35:1 | 0xB4 | Key = guest\_data, Length = 4 |
| 36:4 |  | 4 bytes. Unsigned Integer. Represents Guest Key stored in little-endian form. |
| 40:1 | 0xD2 | Key = firmware\_version, Length = 2 |
| 41:2 |  | 2 bytes. Represents the firmware version. |
| 43:1 | 0xE4 | Key = date\_time, Length = 4 |
| 44:4 |  | 4 bytes. Unsigned Integer. Represents the current date and time set on the lock controller in epoch seconds (seconds elapsed since 00:00:00 UTC on 1 January 1970). |
| 48:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for Get Status request:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**getstatus**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  "status" : “online”,  "battery" : 85,  "deadbolt" : false,  "latch" : false,  "firmware\_version" : <string>,  "date\_time" : "YYYY-MM-DD, hh:mm:ss",  "master\_data" : {  "macrotech\_id" : <unsigned int>,  "hotel\_id" : <unsigned short int>,  "lock\_type" : "guest\_room" or "common\_door",  "floor" : 0 to 255,  "room\_number" : <unsigned short>,  "autolock" : true/false  },  "admin\_key" : <unsigned int>,  "staff\_key" : <unsigned int>,  "guest\_key" : <unsigned int>  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the response corresponds to that lock.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The response object contains multiple fields:   * *status* - can be either “online” or “offline” indicating the lock status * *battery* - provides the battery level in % ranging between 0 and 100. * *deadbolt* - boolean value of true (if engaged) or false (if disengaged) * *latch* - boolean value of true (if closed) or false (if opened) * *firmware\_version* - the version string in “<major>.<minor>.<maintenance>” form (for example: 1.2.40) is sent. * date\_time - the current date and time set on that lock controller, the data is in “YYYY-MM-MM, hh:mm:ss” form * RFID details: Master Data { Macrotech ID, Hotel ID, Lock type, Floor, Room number and Autolock setting (true/false)}, Admin Key {key}, Staff Key {key}, Guest Key {key |

### Get Status from All Lock Controllers

Cloud IoT Server initiates this operation to get status from all lock controllers.

| **MQTT Topic** | {Gateway ID}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**getstatus**”, “Seq” : <integer> } |
| **Description** | In the MQTT topic, **{Gateway ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: Macrotech\_GW\_1) so that the command is broadcasted to all lock controllers in the mesh network associated with the gateway.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  **NOTE:**   * When this command is sent to the gateway, the gateway shall take care of broadcasting this command to all lock controllers in the given mesh network. * CloudExt IoT broker/server would receive as many responses as the number of active lock controllers in the given mesh network. All such responses carry the same Seq number sent in the command request. |

### PDU Structure - IoT Gateway to all Lock controller nodes

IoT Gateway sends the PDU as given in [Get Status - Individual Lock](#_wb3evhharqgh) to all lock controllers.

## Response for Get Status From All Lock Controllers

### PDU Structure - Lock controller node to IoT Gateway

Each lock controller node sends the PDU as given in [Response for Get Status - Individual Lock](#_u0n591h1njzo).

### JSON Payload - IoT Gateway to Cloud IoT Server

The MQTT response message shown in [Get Status - Individual Lock](#_4a8d5dudq0g) is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network.

## Unlock Door - Normal Operation

Cloud IoT Server initiates this operation of unlocking a specific door, where the corresponding deadbolt is already disengaged (normal state).

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**unlock**”, “Seq” : <integer> } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the corresponding door should be unlocked.  The Seq field contains an integer that should be returned back as is in the response payload. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller node for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x11 | Key = lock, Length = 1 |
| 6:1 | 0x00 | 0x0 means unlock the door. |
| 7:1 | 0x00 | End of Packet Marker |

## Response for Unlock Door - Normal Operation

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following response for [*Unlock Door - Normal Operation*](#_bbveaii1nn1t) request on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the [Unlock Door - Normal Operation](#_bbveaii1nn1t) request message. |
| 5:1 | 0x11 | Key = lock, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed unlocking operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for [*Unlock Door - Normal Operation*](#_bbveaii1nn1t) request:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**unlock**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the response corresponds to that lock.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The response object contains a status field, which can be either “success” or “failed”. |

## Unlock Door - Emergency Operation

Cloud IoT Server initiates this operation of unlocking a specific door, where the corresponding deadbolt is engaged (emergency state).

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**emunlock**”, “Seq” : <integer> } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the corresponding door should be unlocked under emergency.  The Seq field contains an integer that should be returned back as is in the response payload. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller node for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x21 | Key = emergency-lock, Length = 1 |
| 6:1 | 0x00 | 0x0 means unlock the door. |
| 7:1 | 0x00 | End of Packet Marker |

## Response for Unlock Door - Emergency Operation

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following response for [*Unlock Door - Emergency Operation*](#_80gu1mt8ta3j) request on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the [Unlock Door - Emergency Operation](#_80gu1mt8ta3j) request message. |
| 5:1 | 0x21 | Key = emergency-lock, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed emergency unlocking operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for [*Unlock Door - Emergency Operation*](#_80gu1mt8ta3j) request:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**emunlock**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the response corresponds to that lock.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The response object contains a status field, which can be either “success” or “failed”. |

## Enable Autolock

Cloud IoT Server initiates this operation to enable autolock on a lock controller.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device ID}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**enable\_autolock**”, “Seq” : <integer> } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the autolock for the corresponding door should be enabled. When autolock is enabled, the locked door can only be opened either using NFC card or via web interface.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller node for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x51 | Key = autolock, Length = 1 |
| 6:1 | 0x01 | 0x01 means to enable autolock. |
| 7:1 | 0x00 | End of Packet Marker |

## Response for Enable Autolock

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following response for [*Enable Autolock*](#_xm4s41bietau) request on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the [Enable Autolock](#_xm4s41bietau) request message. |
| 5:1 | 0x51 | Key = autolock, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed unlocking operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for [*Enable Autolock*](#_xm4s41bietau) request:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**enable\_autolock**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the response corresponds to that lock.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The response object contains a status field, which can be either “success” or “failed”. |

## Disable Autolock

Cloud IoT Server initiates this operation to disable autolock on a lock controller.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device ID}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**disable\_autolock**”, “Seq” : <integer> } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the autolock for the corresponding door should be disabled. When autolock is disabled, the door remains unlocked.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller node for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x51 | Key = autolock, Length = 1 |
| 6:1 | 0x01 | 0x00 means to disable autolock. |
| 7:1 | 0x00 | End of Packet Marker |

## Response for Disable Autolock

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following response for [*Disable Autolock*](#_wix384xrzepr) request on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the [Disable Autolock](#_wix384xrzepr) request message. |
| 5:1 | 0x51 | Key = autolock, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed unlocking operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for [*Disable Autolock*](#_wix384xrzepr) request:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**disable\_autolock**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the response corresponds to that lock.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The response object contains a status field, which can be either “success” or “failed”. |

### Disable Autolock On All Lock Controllers - Emergency Operation

Cloud IoT Server initiates this operation to disable autolock on all lock controllers during emergency.

| **MQTT Topic** | {Gateway ID}/CMD |
| --- | --- |
| **JSON Payload** | { “Command” : “**disable\_autolock**”, “Seq” : <integer> } |
| **Description** | In the MQTT topic, **{Gateway ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: Macrotech\_GW\_1) so that the command is broadcasted to all lock controllers in the mesh network associated with the gateway.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  **NOTE:**   * When this command is sent to the gateway, the gateway shall take care of broadcasting this command to all lock controllers in the given mesh network. * CloudExt IoT broker/server would receive as many responses as the number of active lock controllers in the given mesh network. All such responses carry the same Seq number sent in the command request. |

### PDU Structure - IoT Gateway to all Lock controller nodes

IoT Gateway sends the PDU as given in [PDU Structure - IoT Gateway to a specific Lock controller node](#_v7vyhtuu5283) to all lock controller nodes in the mesh network. It should be noted that the same sequence number is used in the PDU.

## Response for Disable Autolock On All Lock Controllers - Emergency Operation

### PDU Structure - Lock controller node to IoT Gateway

Each lock controller node sends the PDU as given in [Response for Disable Autolock](#_om67revsr1sk).

### JSON Payload - IoT Gateway to Cloud IoT Server

The MQTT response message shown in [Disable Autolock - Individual Lock](#_xl74zrhvcsh0) is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network.

## Set Master Data

Cloud IoT Server initiates this operation by sending a command to set the master data on a lock in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**set\_master\_data**",  "Seq" : <integer>,  "Params" : {  "macrotech\_id" : <unsigned int>,  "hotel\_id" : <unsigned short int>,  "lock\_type" : "guest\_room" or "common\_door",  "floor" : 0 to 255,  "room\_number" : <unsigned short>,  "autolock" : true/false,  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the master data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) Macrotech ID, which is a common ID for all Macrotech’s lock controllers deployed in various locations.  (2) Hotel ID - A unique ID representing a particular hotel  (3) Lock Type indicating if the lock controller is located in a guest room’s door or common door  (4) Floor number (from 0 to 255)  (5) Room Number/Common Door Number  (6) Autolock Setting - true/false |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x8B | Key = master\_data, Length = 11 (B) |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents Macrotech ID stored. |
| 10:2 |  | 2 bytes. Unsigned Short Integer. Represents Hotel ID stored in little-endian form. |
| 12:1 |  | 1 byte. Represents the lock type.  0x01 means the lock is associated with a guest room.  0x02 means the lock is associated with a common door. |
| 13:1 |  | 1 byte. Represents the floor from 0 to 255 (0xFF). |
| 14:2 |  | 2 bytes. Unsigned Short Integer. Represents room number (if lock type is guest room) or common door (if lock type is common door) stored in little-endian form. |
| 16:1 |  | 1 byte. Represents autolock setting (true/false). |
| 17:1 | 0x00 | End of Packet Marker |

## Response for Set Master Data

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Master Data request message. |
| 5:1 | 0x81 | Key = master\_data, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**set\_master\_data**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

## Set Admin Key

Cloud IoT Server initiates this operation by sending a command to set the admin key on a lock in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**set\_admin\_key**",  "Seq" : <integer>,  "Params" : {  "key" : <unsigned int>  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the staff data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) key assigned to admin cards that can be used to unlock all rooms/doors. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0x94 | Key = admin\_key, Length = 4 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents the admin key stored in little-endian form. |
| 10:1 | 0x00 | End of Packet Marker |

## Response for Set Admin Key

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Admin Key request message. |
| 5:1 | 0x91 | Key = admin\_key, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**set\_admin\_key**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

## Set Admin Key - All Lock Controllers

Cloud IoT Server initiates this operation by sending a command to set the admin key on all lock controllers in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Gateway Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**set\_admin\_key**",  "Seq" : <integer>,  "Params" : {  "key" : <unsigned int>  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the staff data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) key assigned to admin cards that can be used to unlock all rooms/doors.  **NOTE:**   * When this command is sent to the gateway, the gateway shall take care of broadcasting this command to all lock controllers in the given mesh network. * CloudExt IoT broker/server would receive as many responses as the number of active lock controllers in the given mesh network. All such responses carry the same Seq number sent in the command request. |

### PDU Structure - IoT Gateway to all Lock controller nodes

IoT Gateway sends the PDU as given in **Set Admin Key for Individual Lock** is sent to all lock controller nodes in the mesh network. It should be noted that the same sequence number is used in the PDU.

## Response for Set Admin Key on All Lock Controllers

### PDU Structure - Lock controller node to IoT Gateway

Each lock controller node sends the PDU as given in **Set Admin Key for Individual Lock**.

### JSON Payload - IoT Gateway to Cloud IoT Server

The MQTT response message shown **Set Admin Key for Individual Lock** is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network.

## Set Staff Key

Cloud IoT Server initiates this operation by sending a command to set the staff key on a lock in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**set\_staff\_key**",  "Seq" : <integer>,  "Params" : {  "key" : <unsigned int>  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the staff data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) key assigned to a staff card that can be used to unlock certain guest room doors. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0xA4 | Key = staff\_key, Length = 4 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents the staff key stored in little-endian form. |
| 10:1 | 0x00 | End of Packet Marker |

## Response for Set Staff Data

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Staff Key request message. |
| 5:1 | 0xA1 | Key = staff\_key, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**set\_staff\_key**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

## Set Guest Key

Cloud IoT Server initiates this operation by sending a command to set the guest data on a lock in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**set\_guest\_key**",  "Seq" : <integer>,  "Params" : {  "key" : <unsigned int>  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the guest data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) key assigned to a guest card to open that particular guest room. To reset the guest key, the key should be set to 0. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0xB4 | Key = guest\_key, Length = 4 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents the guest key stored in little-endian form. |
| 10:1 | 0x00 | End of Packet Marker |

## Response for Set Guest Key

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Guest Key request message. |
| 5:1 | 0xB1 | Key = guest\_key, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**set\_guest\_key**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

## Delete RFID Data

Cloud IoT Server initiates this operation by sending a command to delete/erase specific type (master/staff/guest/all) of RFID card data stored in a lock controller.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Device Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**deletecard**",  "Seq" : <integer>,  "Params" : {  "card\_id" : <unsigned int>,  "card\_type" : "admin" | "staff" | "guest" | "all"  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the RFID data should be deleted from that lock controller.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following mutually exclusive parameters:   1. Card id, the id of the card to be deleted. If the card id field is present, the card type field is ignored. 2. Card type, the type of card to be deleted. Card type can be either one of these: admin, staff, guest, all. If all is set, all card data stored on the specified lock controller shall be deleted. Note that if the card id field is present, then this field is ignored. |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is **0x09**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the **Seq** field. |
| 5:1 | 0xC5 | Key = delete\_rfid\_data, Length = 5 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents Card ID (Admin/Staff/Guest) stored in little-endian form. |
| 10:1 |  | 1 Byte. Represents the card type.  0xFF - All cards (Admin, Staff, Guest)  0x1 - Admin card  0x2 - Staff card  0x3 - Guest card |
| 11:1 | 0x00 | End of Packet Marker |

## Response for Delete RFID Data

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is **0x19**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Staff Data request message. |
| 5:1 | 0xB1 | Key = delete\_rfid\_data, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| **MQTT Topic** | {Device Id}/CMD\_RESP |
| --- | --- |
| **JSON Payload** | {  “Command” : “**deletecard**”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| **Description** | In the MQTT topic, **{Device ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

### Delete RFID Card Data from All Lock Controllers

Cloud IoT Server initiates this operation to delete RFID card data from all lock controllers.

### JSON Payload - Cloud IoT Server to IoT Gateway

| **MQTT Topic** | {Gateway Id}/CMD |
| --- | --- |
| **JSON Payload** | {  "Command" : "**deletecard**",  "Seq" : <integer>,  "Params" : {  "card\_id" : <unsigned int>,  "card\_type" : "admin" | "staff" | "guest" | "all"  }  } |
| **Description** | In the MQTT topic, **{Gateway ID}** is set with the user-readable string as listed in CloudExt’s UI (for example: Macrotech\_GW\_1) so that the command is broadcasted to all lock controllers in the mesh network associated with the gateway.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following mutually exclusive parameters:   1. Card id, the id of the card to be deleted. If the card id field is present, the card type field is ignored. 2. Card type, the type of card to be deleted. Card type can be either one of these: admin, staff, guest, all. If all is set, all card data stored on the specified lock controller shall be deleted. Note that if the card id field is present, then this field is ignored.   **NOTE:**   * When this command is sent to the gateway, the gateway shall take care of broadcasting this command to all lock controllers in the given mesh network. * CloudExt IoT broker/server would receive as many responses as the number of active lock controllers in the given mesh network. All such responses carry the same Seq number sent in the command request. |

**NOTE 1:** For PDU structure sent between Gateway to Lock Controller and back to Gateway, refer the corresponding sections under “Delete RFID Card Data”.

**NOTE 2:** The JSON Payload sent from IoT Gateway to Cloud IoT Server is similar to that sent out in case of “Delete RFID Card Data”.

## Set Date and Time

Cloud IoT Server initiates this operation by sending a command to set the current date and time on a lock controller in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| MQTT Topic | {Device Id}/CMD |
| --- | --- |
| JSON Payload | {  "Command" : "set\_date\_time",  "Seq" : <integer>,  "Params" : {  "date\_time" : "YYYY-MM-DD, hh:mm:ss"  }  } |
| Description | In the MQTT topic, {Device ID} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the staff data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  The params field contains the following parameters:  (1) date\_time, a string in “YYYY-MM-DD, hh:mm:ss” form indicating the date and time to be set on the lock controller. The date\_time fields are as listed below:  YYYY - Year (for example: 2022)  MM - Month in the range 01 to 12 (01 - Jan … 12 - Dec)  DD - Date in the range 01 to 31  hh - Hour in 24-hour format in the range 00 to 23  mm - Minutes in the range 00 to 59  ss - Seconds in the range 00 to 59 |

### PDU Structure - IoT Gateway to a specific Lock controller node

The following PDU is sent to Lock controller nodes for Set-State Endpoint (EP), which is 0x09:

| Byte:Length | Field Value | Description |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This is actually the value received in the request payload for the Seq field. |
| 5:1 | 0xE4 | Key = admin\_key, Length = 4 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents the date and time in epoch seconds (seconds elapsed since 00:00:00 UTC on 1 January 1970). |
| 10:1 | 0x00 | End of Packet Marker |

## Response for Set Date and Time

### PDU Structure - Lock controller node to IoT Gateway

The following PDU is sent from the Lock controller node on Acknowledgement-State Endpoint (EP) for the destination, which is 0x19:

| Byte:Length | Field Value | Description |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. Signed int (4 bytes). This request id must be the request id that was sent in the Set Admin Key request message. |
| 5:1 | 0xE1 | Key = date\_time, Length = 1 |
| 6:1 | 0x0 or 0x1 | Error code. 0x0 means a successful operation. 0x1 indicates a failed operation. |
| 7:1 | 0x00 | End of Packet Marker. |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT response message is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network:

| MQTT Topic | {Device Id}/CMD\_RESP |
| --- | --- |
| JSON Payload | {  “Command” : “set\_date\_time”,  “Seq” : <integer that was received in the request message>,  “Response” : {  “status” : “success”  }  } |
| Description | In the MQTT topic, {Device ID} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the JSON payload is coming from that lock controller.  The JSON payload contains three fields: command, seq, and response.  The command field contains the name of the command for which the response is associated.  The seq (sequence) field contains the integer that was received in the corresponding command request’s seq field.  The status field in the response can be either “success” or “failed” |

## Set Date and Time - All Lock Controllers

Cloud IoT Server initiates this operation by sending a command to set the current date and time on all lock controllers in the Wirepas RF mesh network to which the gateway is connected.

### JSON Payload - Cloud IoT Server to IoT Gateway

| MQTT Topic | {Gateway Id}/CMD |
| --- | --- |
| JSON Payload | {  "Command" : "set\_date\_time",  "Seq" : <integer>,  "Params" : {  "date\_time" : "YYYY-MM-DD, hh:mm:ss"  }  } |
| Description | In the MQTT topic, {Device ID} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the staff data for the corresponding lock controller should be set.  The Seq field contains an integer that should be returned back as is in the response payload. The Seq id is used by the IoT server for tracking the requests.  (1) date\_time, a string in “YYYY-MM-DD, hh:mm:ss” form indicating the date and time to be set on the lock controller. The date\_time fields are as listed below:  YYYY - Year (for example: 2022)  MM - Month in the range 01 to 12 (01 - Jan … 12 - Dec)  DD - Date in the range 01 to 31  hh - Hour in 24-hour format in the range 00 to 23  mm - Minutes in the range 00 to 59  ss - Seconds in the range 00 to 59  NOTE:   * When this command is sent to the gateway, the gateway shall take care of broadcasting this command to all lock controllers in the given mesh network. * CloudExt IoT broker/server would receive as many responses as the number of active lock controllers in the given mesh network. All such responses carry the same Seq number sent in the command request. |

### PDU Structure - IoT Gateway to all Lock controller nodes

IoT Gateway sends the PDU as given in Set Date and Time on Individual Lock is sent to all lock controller nodes in the mesh network. It should be noted that the same sequence number is used in the PDU.

## Response for Set Date and Time on All Lock Controllers

### PDU Structure - Lock controller node to IoT Gateway

Each lock controller node sends the PDU as given in Set Date and Time on Individual Lock.

### JSON Payload - IoT Gateway to Cloud IoT Server

The MQTT response message shown Set Date and Time on Individual Lock is sent from IoT Gateway to Cloud IoT Server for every response received from the lock controllers in the mesh network.

## Heartbeat - Periodic Telemetry Message

All lock controllers in the wirepas RF mesh network periodically (frequency is configurable) send out the heartbeat message, which is sent to Cloud IoT Server for telemetry purpose.

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following heartbeat message to IoT Gateway on Notification-Telemetry Endpoint (EP) for the destination, which is **0x12**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. For heartbeat messages, the request must be set to -1. |
| 5:1 | 0x11 | Key = status, Length = 1 |
| 6:1 | 0x0 or 0x1 | Status. 0x0 means offline and 0x1 means online. |
| 7:1 | 0x21 | Key = battery, Length = 1 |
| 8:1 | 0x0 to 0x64 | Battery level in percentage. 0 (0x) to 100 (0x64). |
| 9:1 | 0x31 | Key = deadbolt, Length = 1 |
| 10:1 | 0x0 or 0x1 | Deadbolt status. 0x0 means disengaged and 0x1 means engaged. |
| 11:1 | 0x41 | Key = latch, Length = 1 |
| 12:1 | 0x0 or 0x1 | Latch status. 0x0 means opened and 0x1 means closed. |
| 13:1 | 0x51 | Key = autolock, Length = 1 |
| 14:1 | 0x0 or 0x1 | Autolock status. 0x0 means disabled and 0x1 means enabled. |
| 15:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT message is sent from IoT Gateway to Cloud IoT Server for heartbeat message:

| **MQTT Topic** | {Device Id}/NOTIFICATION |
| --- | --- |
| **JSON Payload** | {  “status” : “online”,  “battery” : 85,  “deadbolt” : false,  “latch” : false,  “autolock” : false  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the notification corresponds to that lock.  The heartbeat contents are part of Notification object comprising of the following fields:   * *status* - can be either “online” or “offline” indicating the lock status * *battery* - provides the battery level in % ranging between 0 and 100. * *deadbolt* - boolean value of true (if engaged) or false (if disengaged) * *latch* - boolean value of true (if closed) or false (if opened) * *autolock* - boolean value of true (if enabled) or false (if disabled) |

## Status Update Message

A lock controller sends out a *Status Update* message, when it detects any changes in the state of the following parameters:

* Lock
* Deadbolt
* Latch

The PDU structure and the JSON payload are the same as that sent out in case of Hearbeat message.

### PDU Structure - Lock controller node to IoT Gateway

See the PDU structure described in [Heartbeat - Periodic Telemetry Message](#_546e1y1vcbfy) section.

### JSON Payload - IoT Gateway to Cloud IoT Server

See the JSON payload described in [Heartbeat - Periodic Telemetry Message](#_o2ttv01k3imv) section.

## Alarm - Low-Battery

Lock controllers emit low-battery alarm/alert when the battery voltage level drops below the minimum operating value

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following alarm message to IoT Gateway on Notification-Alarm Endpoint (EP) for the destination, which is **0x13**:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. For alarm messages, the request must be set to -1. |
| 5:1 | 0x11 | Key = low-battery, Length = 1 |
| 6:1 | 0x1 | The value is always 0x1 |
| 7:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT message is sent from IoT Gateway to Cloud IoT Server for alarms:

| **MQTT Topic** | {Device Id}/ALARM |
| --- | --- |
| **JSON Payload** | {  “low-battery” : true  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the alarm corresponds to that lock.  In the JSON payload’s Notification field, low-battery is always true indicating the battery voltage level has dropped below the minimum operating value. |

## Alarm - Deadbolt Alert

Lock controller emits deadbolt alarm/alert when the corresponding deadbolt is either engaged or disengaged.

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following alarm message to IoT Gateway on Notification-Alarm Endpoint (EP) for the destination, which is 0x13:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. For alarm messages, the request must be set to -1. |
| 5:1 | 0x21 | Key = deadbolt-alert, Length = 1 |
| 6:1 | 0x0 or 0x1 | 0x1 means the deadbolt is engaged and 0x0 means the deadbolt is disengaged. |
| 7:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT message is sent from IoT Gateway to Cloud IoT Server for alarms:

| **MQTT Topic** | {Device Id}/ALARM |
| --- | --- |
| **JSON Payload** | {  “deadbolt-alert” : true  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the alarm corresponds to that lock.  In the JSON payload’s Notification field, deadbolt-alert can be either true (deadbolt engaged) or false (deadbolt disengaged). |

## Alarm - Latch Alert

Lock controller emits latch alarm/alert when the corresponding latch is either closed or opened.

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following alarm message to IoT Gateway on Notification-Alarm Endpoint (EP) for the destination, which is 0x13:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. For alarm messages, the request must be set to -1. |
| 5:1 | 0x31 | Key = latch-alert, Length = 1 |
| 6:1 | 0x0 or 0x1 | 0x1 means the latch is closed and 0x0 means the latch is opened. |
| 7:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT message is sent from IoT Gateway to Cloud IoT Server for alarms:

| **MQTT Topic** | {Device Id}/ALARM |
| --- | --- |
| **JSON Payload** | {  “latch-alert” : true  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the alarm corresponds to that lock.  In the JSON payload’s Notification field, latch-alert can be either true (latch closed) or false (latch opened). |

## Alarm - Unlock by RFID Card Alert

Lock controller emits unlock by RFID card alert when an attempt to unlock the door using a RFID card occurs.

### PDU Structure - Lock controller node to IoT Gateway

The lock controller node sends the following alarm message to IoT Gateway on Notification-Alarm Endpoint (EP) for the destination, which is 0x13:

| **Byte:Length** | **Field Value** | **Description** |
| --- | --- | --- |
| 1:4 | <Req Id> | Request ID. For alarm messages, the request must be set to -1. |
| 5:1 | 0x46 | Key = unlock-by-rfid-alert, Length = 6 |
| 6:4 |  | 4 bytes. Unsigned Integer. Represents card ID stored in little-endian form. |
| 10:1 |  | 1 byte. Represents the card type. 0x01 (master card) or 0x02 (staff card) or 0x03 (guest card) |
| 11:1 |  | 1 byte. Represents the status - whether the unlock operation was successful (0x01) or failed (0x0). |
| 12:1 | 0x00 | End of Packet Marker |

### JSON Payload - IoT Gateway to Cloud IoT Server

The following MQTT message is sent from IoT Gateway to Cloud IoT Server for alarms:

| **MQTT Topic** | {Device Id}/ALARM |
| --- | --- |
| **JSON Payload** | {  “unlock-by-rfid-alert” : {  “card\_id” : <unsigned int>,  “card\_type” : “admin” | “staff” | “guest”,  “status” : “success” | “failed”  }  } |
| **Description** | In MQTT topic, {Device Id} is set with the user-readable string as listed in CloudExt’s UI (for example: F1-R101-Lock), indicating that the alarm corresponds to that lock.  In the JSON payload’s Notification field contains the following data:   * Card ID * Card Type (string: admin, staff or guest) * Status (string: success or failed) |

# Appendix A - Wirepas Node Address and Device Id mapping CSV file

Here is an example file showing the mapping of Wirepas Node Address and Device Id:

| name,address,type,remarks  F1-R101-Lock,0xDFD81E,Lock,"Lock in 1st Floor, Room 101"  F8-R812-Lock,0xB56FCD,Lock,"Lock in 8th Floor, Room 812"  F1-R102-Lock,0x38E823,Lock,"Lock in 1st Floor, Room 102" |
| --- |

The mapping CSV file contains the following columns/fields:

* name - The name of the lock controller as identified in CloudExt IoT Platform UI. It corresponds to **Device Id**. It is expected to be in human-readable format, for example, F8-R812-Lock, representing the lock in Room 812 on the 8th floor.
* address - The address assigned to the lock controller in Wirepas RF mesh network.
* type - Type of the node. For lock controllers, it should be **Lock**.
* remarks - Space for writing free form text, which can be used, if required, to include additional information. This field is optional.

# Appendix B - Persistent Storage in Lock Controller for storing Master, Staff and Guest Data

The following table provides various parameters along with their size associated with Master Data, Staff Data and Guest Data that are stored in the persistent storage of each lock controller. Those data persisted in non volatile memory shall be used for validating RFID access cards when Wirepas mesh network is down.

| **Parameter** | **Size (Bytes)** | **Type** | **Description** |
| --- | --- | --- | --- |
| Macrotech ID | 4 | Unsigned Int | Macrotech ID. Common for all lock controllers from Macrotech used across various hotels. |
| Hotel ID | 2 | Unsigned Short Int | Hotel ID that can range from 0 to 65535 used to uniquely identify a particular hotel. |
| Master ID | 4 | Unsigned Int | The unique key that is stored in the Master/Admin card. It is assumed that there is only one master/admin card per hotel. |
| Guest ID | 4 | Unsigned Int | The unique key that is stored in a Guest card given to the guest for a specific guest room. In case of lock controllers in common doors, this value is stored as 0. |
| Lock Type | 1 | Byte | Used for identifying the lock type. 0x01 means the lock is associated with a guest room and 0x02 means it is for a common door including the lift. |
| Floor | 1 | Byte | 0 - 255 indicating the floor number where the lock controller is located. |
| Room Number/  Common Door | 2 | Unsigned Short Int | If Lock type is of type 'guest room' (0x01), this value contains the room number or else, the common door. |
| Staff ID | 4 | Unsigned Int | The unique key that is stored in a Staff card given to the staff for that guest room. Currently, only one card per guest room is supported. |

# 