OrionM2M

Software

OrionNetworkServer

Description of WEBSOCKET API

- Connection
- Commands

building
connected future

www.orion-m2m.com

LoRa Alliance Member™

Contents

INTRODUCTION 3
1. TERMS AND DEFINITIONS 4
2. CONNECTION 5
3. COMMANDS 6
3.1. Get a list of connected clients 7
3.2. Disconnect client 8
3.3. Get applications 9
3.4. Create application 10
3.5. Delete application 11
3.6. Get a list of tariffs 12
3.7. Get devices 13
3.8. Create device 15
3.9. Update device 18
3.10. Delete device 21
3.11. Subscribe to realtime data 22
3.12. Unsubscribe to realtime data 24
3.13. Checking the integrity of the connection 25
3.14. Change timeout of checking the connection integrity 26
3.15. Select data from database 27
3.16. Downlink - get LoRA WAN Device Class 29
3.17. Downlink - get the maximum data size at the time of the last Uplink 31
3.18. Downlink – send data to device 33
3.19. Downlink - confirmation of sending to device 35
3.20. Get sent data from database 36
3.21. Get device coordinates 38
3.22. Set coordinates to device 40
3.23. Additional commands 41
4. REVISION SHEET 42

INTRODUCTION

This manual contains information on the exchange protocol between NS and APP.

1. TERMS AND DEFINITIONS

Activation by Personalization - activation by pre-recording personal settings in the device, before connecting the device to the network;

AppEUI - (8-BYTE, EUI64) globally unique application identifier for routing the received data by the Network Server to the application server (AppServer);

AppKey - unique (16- BYTE, AES-128) an encryption key generated by the AppServer for this particular device;

AppServer – application server of client;

DevEUI - (8- BYTE, EUI64) globally unique devise identifier (End-device identifier). It can be assigned by the device manufacturer (by analogy with the MAC address), in a limited amount it can be obtained from the available pool of operator identifiers, or received by the owner of the node in the IEEE pool;

DevAddr — 32-bit (4-BYTE) a network address for addressing packets at the network layer. It has a unique value within the operator's network (an analogy with a MAC address, which also provides addressing at the 2nd level of the OSI model in Ethernet networks, but the method of obtaining DevAddr with OTAA is similar to obtaining a dynamic IP address, received from a DHCP server in TCP/IP networks). The upper 7 bits of DevAddr contain the operator's network address NwkID, this value must be unique for nearby networks and for networks with overlapping coverage areas. Most often, to designate DevAddr, a four-byte sequence is used (for example: 02:D1:D2:01), in which the high byte is the address of the NwkID network;

End-node – end user device on the LoRaWAN™ network;

LoRa® - physical layer protocol describing the connection in LoRaWAN™ networks;

Long Range wide-area networks (LoRaWAN)– a long-range global network allows you to connect devices with a battery life of up to ten years, transmit data over long distances. Unlike mobile networks, LoRaWAN ™ networks are hybrid and can be either private or public:

Network ID (NwkID) - 7-bit network identifier, which is part of the network address (DevAddr), is used to identify networks operating in the same territory;

NwkSKey – network session key [128-bit], used for calculating and checking the MIC (message integrity code) field of messages during the exchange between the end-node and the Network Server, as well as for encrypting MAC-level messages;

AppSKey - session key [128-bit] used to encrypt data at the application level (between the terminal device and the application server);

Over-the-Air Activation (OTAA) – over the air activation, a method of activating devices in the LoRaWAN ™ network through the provider's network at the moment they are turned on;

Activation by Personalization (ABP) - This is a method of activating by personalization devices in the LoRaWAN ™ network;

APP_ID - API application id;

APP_KEY - API application key.

2. CONNECTION

To complete the connection procedure, enter the command:

wss://iot.orion-m2m.com/ws/v2/?app_id=APP_ID&token=APP_KEY

Answer:

```
{
    "cmd": "connected",
    "status": uint8,                    // 1 – OK; 2 - ERROR
    "error_code":  uint8,               // 1 - Auth error,
                                        // 2 - Clients count limit reached
    "clients_count": uint8,
    "consumer_count": uint8
}
```

3. COMMANDS

There are commands which allow performing:

- Get applications;
- Create application;
- Delete application;
- Get a list of tariffs;
- Get devices;
- Create device;
- Update device;
- Delete device;
- Subscribe to realtime data;
- Unsubscribe to realtime data;
- Checking the integrity of the connection;
- Select data from database;
- Downlink – get LoRA WAN Device Class
- Downlink – get the maximum data size at the time of the last Uplink
- Downlink – send data to device;
- Downlink - confirmation of sending to device;
- Get sent data from database;
- Get device coordinates;
- Set coordinates to device;
- Additional commands.

Below are the commands of each item.

3.1. Get a list of connected clients

Request:

```
{
"cmd": "connections_list",          // Optional
"operation_id": string,
}
```

Example request:

```
{
   "cmd": "connections_list",
   "operation_id": "abc",
}
```

Accept:

```
{
   "cmd": "connections_list",
   "operation_id": string,
   "status": uint8,                 // 1 - OK, 2 - ERROR
   "error_code": uint8,
   "connections": [
   "applicationCount" : uint32,
   "applicationList": [
   {
   "peer": string,
   "uuid":  string,
   "consume": bool
   }
   ]
}
```

Sample accept:

```
{
   "cmd": "get_applications_resp",
   "operation_id": "abc",
   ""status": 1,
   " connections": [
   {
   "peer": "tcp4:172.20.0.1:56758",
            "uuid":  "80f0769cae484a4ea7a22a023f2f6379",
            "consume": true
   },
   {
   "peer": "tcp4:172.20.0.1:56770",
   "uuid":  "437dc414577d4efda7121e78849bfa7a",
   "consume": true
   },
   ]
   }
```

3.2. Disconnect client

Request:

```
{
    "cmd": "connection_kill",
    "operation_id": string,          //Optional
    "uuid": string,                  // Connection UUID

}
```

Example request:

```
{
    "cmd": "connection_kill",
    "operation_id": "abc",
    "uuid":
    "80f0769cae484a4ea7a22a023f2f6379"

}
```

Accept:

```
{
    "cmd": "connection_kill_resp",
    "operation_id": string,
    "status": uint8,                 // 1 - OK, 2 - ERROR
    "error_code": uint8              // 1 - Connection UUID not found
}
```

Sample accept:

```
{
    "cmd": "connection_kill_resp",
    "operation_id": "abc",
    "status": 1
}
```

3.3. Get applications

Request:

```
{
    "cmd": "get_applications_req",
    "operation_id": string,              //Optional
    "appEui": string,                    // Optional
}
```

Example request:

```
{
    "cmd": "get_applications_req",
    "operation_id": “abc”,
    "appEui": "819EA8554FCD762A"
}
```

Accept:

```
{
    "cmd": "get_applications_resp",
    "operation_id": string,
    "status": uint8,                     // 1 - OK, 2 - ERROR
    "error_code": uint8,                 // 1 - Unauthorized access
                                         // 2 - Application not found
    "applicationCount" : uint32,         // 3 - Unexpected error
    "applicationList": [
        {
            "appEui": string,
            "tokens": [ string, ]
        }
    ]
}
```

Sample accept:

```
{
    "cmd": "get_applications_resp",
    "operation_id": "abc",
    "applicationCount": 2,
    "applicationList": [
        {
            "appEui": "6E5966FEE2380BB2",
            "tokens": [
                "1VZYsWzJeyDIUHSmwddlbD",
                "YrLNyNnPNW3bYhoOHwtEN8"
            ]
        },
    ]
}
```

3.4. Create application

Request:

```
{
"cmd": "create_application_req",
"operation_id": string,                    // Optional
"appName": string
}
```

Example request:

```
{
"cmd": "create_application_req",
"operation_id": “abc”,
"appName": "testApp"
}
```

Accept:

```
{
"cmd": "create_application_resp",
"operation_id": string,
"status": uint8,                           // 1 – OK; 2 - ERROR
"error_code": uint8,                       // 1 - Unauthorized access
                                           // 2 - Required field is not filled
“app_id”: string,                          // 3 - Unexpected error
“app_eui”: string,
“token”: string
}
```

Sample accept:

```
{
"cmd": "create_application_resp",
"operation_id": “abc”,
"status": 1,
“app_id”: “358098F3”,
“app_eui”: “258BD226017CBF01”,
“token”: “737x6WuNdSBC4VFWhuyHHz”
}
```

3.5. Delete application

Request:

```
{
"cmd": "delete_application_req",
"operation_id": string,                    // Optional
"appEui": string
}
```

Example request:

```
{
"cmd": "delete_application_req",
"operation_id": “cdma”,
"appEui": “258BD226017CBF01”
}
```

Accept:

```
{
"cmd": "delete_application_resp",
"operation_id": string,
"status": uint8,                           // 1 – OK; 2 - ERROR
"error_code": uint8                        // 1 -Unauthorized access
}                                          // 2 - appEui is required
                                           // 3 - Application not found
                                           // 4 - Application used
                                           // 5 - Unexpected error
```

Sample accept:

```
{
"cmd": "delete_application_resp"
"operation_id": "cdma",
"status": 1,
}
```

3.6. Get a list of tariffs

Request:

```
{
"cmd": "get_tariffs_req",
"operation_id": string                    //Optional
}
```

Example request:

```
{
"cmd": "get_tariffs_req",
"operation_id": “cdma”
}
```

Accept:

```
{
"cmd": "get_tariffs_resp",
"operation_id": string,
"status": uint8,                          // 1 – OK; 2 - ERROR
"error_code": uint8                       // 1 - Tariffs not found
}                                         // 2 - Unexpected error
```

Sample accept:

```
{
   "cmd": "get_tariffs_resp",
   "status": 1,
   "operation_id": "cdma",
   "tariffs": [
      {
         "code": "A1",
         "name": "Tariff 1",
         "description": ""
      },
      {
         "code": "A2",
         "name": "Tariff 2",
         "description": ""
      }
   ]
}
```

3.7. Get devices

Request:

```
{
    "cmd": "get_devices_req",
    "operation_id": string,              // Optional
   "devEui": string                      // Optional
}
```

Example request:

```
{
   "cmd": "get_devices_req",
   "operation_id": "Qlg",
   "devEui": "D745F677B2412E03"
}
```

Accept:

```
{
    "cmd": "get_devices_resp",
    "status": uint8,                     // 1 – OK; 2 - ERROR
    "error_code": uint8,                 // 1 - Unauthorized access
                                         // 2 - Required field is not filled
    "deviceCount" : uint32,              // 3 - Data is filled in with an error
        "deviceList": [ {
            "devEui": string,
            "devAddr": string,
            “application_eui”: string,
            “description”: string,
            “app_key”:  string,
            “nwks_key”: string,
            “apps_key”: string,
            “node_class”: uint8,
            “rx_window”: uint8,
            “adr”: bool,
            “adr_interval”: uint8,
            “adr_min”: uint8,
            “adr_max”: uint8,
            “adr_fix”: bool,
            “duty_cycle”: uint8,
            “seq_up”: unit32,
            “seq_down”: unit32,
            “seq_chk”: unit32,
           “region”: uint8,
           “activation”: uint8,
           “lat”: string,
           “lon”: string
        }]
```

Sample accept:


```
{
  "cmd": "get_devices_resp",
  "operation_id": "Qlg",
  "status": 1,
  "deviceCount": 1,
  "deviceList": [
    {
      "devEui": "D745F677B2412E03",
      "devAddr": "D1FD37F0",
      "application_eui": "819EA8554FCD762A",
      "description": "",
      "app_key": "2B7E151628AED2A6ABF7158809CF4F3C",
      "nwks_key": "D745F677B2412E03268C11F3E1B6893C",
      "apps_key": "2D48A7ECFF70AF6323B7A57186C23E11",
      "node_class": 1,
      "rx_window": 1,
      "adr": true,
      "adr_interval": 5,
      "adr_min": null,
      "adr_max": null,
      "adr_fix": null,
      "duty_cycle": 0,
      "seq_up": 0,
      "seq_down": 0,
      "seq_chk": 0,
      "region": 1,
      "activation": 0,
      "lat": 45.234232,
      "lon": 76.342534
    }
  ]
}
```


Note:

Using a **high** access token, the method will return a list of all devices owned by the user.

Using a **low** access token, the method will return a list of devices belonging to a specific application.

3.8. Create device

Request:

```
{
    "cmd":
"create_device_req",              // Optional
    “operation_id": string,       // Necessarily
   "devEui": string,              // Necessarily
    "devAddr": string,            // Necessarily
    “application_eui”: string,    // Optional
    “description”: string,        // Necessarily
    “app_key”:  string,           // Necessarily
    “tariff_code”: string,        // Optional
    “nwks_key”: string,           // Optional
    “apps_key”: string,           // Default: 1    (1-A, 2-B, 3-C)
    “node_class”: uint8,          // Default: 1    (1-RX1, 2-RX2)
     “rx_window”: uint8,          // Default: True
     “adr”: bool,                 // Default: 20
     “adr_interval”: uint8,       // Default: 0    (0 - 7) (DR0-DR7)
     “adr_min”: uint8,            // Default: 5    (0 - 7) (DR0-DR7)
     “adr_max”: uint8,            // Default: NULL    (0 - 7) (DR0-DR7)
     “adr_fix”: bool,             // Default: 0
     “duty_cycle”: uint8,         // Default: 0
     “seq_up”: unit32,            // Default: 0
     “seq_down”: unit32,          // Default: 0    (0-Strict, 1-Relaxed)
     “seq_chk”: bool,             // Default: 6 (KZ865-868)    (0-EU, 1-KZ_OLD, 2-RU, 3-BY,
     “region”: uint8,             4-UA, 5-KG, 6-KZ865-868)
                                  // Default: 0 (0-OTAA, 1-ABP)
     “activation”: uint8,         // Default: 0
     “down_queue”: uint8,         // Default: NULL
     "lat": float,                // Default: NULL
     "lon": float

}
```

Example request:

```
{
   "cmd": "create_device_req",
   "operation_id": "rnt",
   "devEui": "7DA50BE8C74CE6DC",
   "application_eui": "819EA8554FCD762A",
   "app_key": "948205A2939E8C999D641DE5CB731861",
   “tariff_code”: “A1”
}
```

Accept:

```
{
    "cmd":
"create_device_resp",
    "operation_id": string,          // 1 – OK; 2 - ERROR
    "status": uint8,                 // 0 - Unexpected error
    "error_code": uint8,             // 1 - Unauthorized access
                                     // 2 - Required field is not filled
                                     // 3 - Data is filled in with an error
                                     // 4 - Uniq failed
                                     // 5 - Application does not exists
                                     // 6 - Tariff does not exists

    "devEui": string,
    "devAddr": string,
    “application_eui”: string,
    “description”: string,
    “app_key”:  string,
    “nwks_key”: string,
    “apps_key”: string,
    “node_class”: uint8,
     “rx_window”: uint8,
     “adr”: bool,
     “adr_interval”: uint8,
     “adr_min”: uint8,
     “adr_max”: uint8,
     “adr_fix”: bool,
     “duty_cycle”: uint8,
     “seq_up”: unit32,
     “seq_down”: unit32,
     “seq_chk”: bool,
     “region”: uint8,
     “activation”: uint8,
     “down_queue”: uint8,
     “tariff_code”: string,
     "lat": float,
     "lon": float

}
```

Sample accept:

```
{
   "cmd": "create_device_resp",
   "operation_id": "rnt",
   "status": 1,
   "devEui": "7DA50BE8C74CE6DC",
   "devAddr": "41410509",
```

```
    "application_eui": "819EA8554FCD762A",
    "description": "",
    "app_key": "948205A2939E8C999D641DE5CB731861",
    "nwks_key": "EEFBF983C00B35240C9A7C2D8D011301",
    "apps_key": "6F17B97CEA2C63D1089EF4268FD68D38",
    “tariff_code”: “A1”,
    "node_class": 1,
    "rx_window": 1,
    "adr": true,
    "adr_interval": 5,
    "adr_min": null,
    "adr_max": null,
    "adr_fix": null,
    "duty_cycle": 0,
    "seq_up": 0,
    "seq_down": 0,
    "seq_chk": 0,
    "region": 1,
    "activation": 0,
    "down_queue": 0,
    "lat": 42.234,
    "lon": 75.523

}
```

3.9. Update device

Request:

```
{
    "cmd": "update_device_req",
    “operation_id”: string,                 // Optional
"devEui": string,                           // Optional
    "devAddr": string,                      // Optional
    “application_eui”: string,
    “description”: string,
    “app_key”:  string,
    “nwks_key”: string,
    “apps_key”: string,
    “node_class”: uint8,
     “rx_window”: uint8,
     “adr”: bool,
     “adr_interval”: uint8,
     “adr_min”: uint8,
     “adr_max”: uint8,
     “adr_fix”: bool,
     “duty_cycle”: uint8,
     “seq_up”: unit32,
     “seq_down”: unit32,
     “seq_chk”: bool,
     “region”: uint8,
     “activation”: uint8,
     “down_queue”: uint8,
     "lat": float,
     "lon": float

}
```

Example request:

```
{
   "cmd": "update_device_req",
   "operation_id": "rdk",
   "devEui": "7DA50BE8C74CE6DC",
   "application_eui": "6042366ED7A302CF",
   "app_key":
"79AFAC4879C5A1E2BFEDF4A4E35C0C6B",
   "region": 2,
}
```

Accept:

```
{
    "cmd": "update_device_resp",
    “operation_id”: string,
    "status": uint8,                    // 1 – OK; 2 - ERROR
    "error_code": uint8,                // 1 - Unauthorized access
                                        // 2 - Required field is not filled
                                        // 3 - Data is filled in with an error
                                        // 4 - Uniq failed
                                        // 5 - Application does not exists
"devEui": string,                       // 6 - Tariff does not exists
    "devAddr": string,                  // 7 - Tariff can be changed no more than once a day
    “application_eui”: string,          // 8 - Unexpected error
    “description”: string,              // 9 - Node does not exists
    “app_key”:  string,
    “nwks_key”: string,
    “apps_key”: string,
    “node_class”: uint8,
     “rx_window”: uint8,
     “adr”: bool,
     “adr_interval”: uint8,
     “adr_min”: uint8,
     “adr_max”: uint8,
     “adr_fix”: bool,
     “duty_cycle”: uint8,
     “seq_up”: unit32,
     “seq_down”: unit32,
     “seq_chk”: bool,
     “region”: uint8,
     “activation”: uint8,
     “down_queue”: uint8,
     "lat": float,
     "lon": float

}
```

Sample accept:

```
{
   "cmd": "update_device_resp",
   "operation_id": "rdk",
   "status": 1,
   "devEui": "7DA50BE8C74CE6DC",
   "devAddr": "2F1BA78D",
   "application_eui": "6042366ED7A302CF",
   "description": "",
   "app_key": "79AFAC4879C5A1E2BFEDF4A4E35C0C6B",
```

"nwks_key": "FF93AB45868BC0ED652DB9A0698CCAD7",
"apps_key": "B65466247DC7F5B5CCA383AD47233D86",
"node_class": 1,
"rx_window": 1,
"adr": true,
"adr_interval": 5,
"adr_min": null,
"adr_max": null,
"adr_fix": null,
"duty_cycle": 0,
"seq_up": 0,
"seq_down": 0,
"seq_chk": 0,
"region": 2,
"activation": 0,
"down_queue": 0,
"lat": 42.221848,
"lon": 75.898052
}

3.10. Delete device

Request:

```
{
    "cmd": "delete_device",
    "operation_id": string,          // Optional
   "devEui": string,                 // Optional
    "devAddr": string,               // Optional
 }
```

Example request:

```
{
   "cmd": "delete_device_req",
   "operation_id": "rdk",
   "devEui": "7DA50BE8C74CE6DC"
}
```

Accept:

```
{
    "cmd": "delete_device_resp",
    "status": uint8,
    "error_code": uint8,          // 1 – OK; 2 - ERROR
}                                 // 1 - Unauthorized access
                                  // 2 - Required field is not filled
                                  // 3 - Node does not exists
```

Sample accept:

```
{
   "cmd": "delete_device_resp",
   "operation_id": "rdk",
   "status": 1
}
```

3.11. Subscribe to realtime data

Request:

```
{
    "cmd": "consume_req"
}
```

Accept:

```
{
    "cmd": "consume_resp",
    "status": uint8,                    // 1 – OK; 2 - ERROR
    "error_code": uint8,                // 1 - Consumer limit reached,
                                        // 2 - Already consumed,
                                        // 3 - Disabled application
    "consumer_count": uint8
}
```

Data:

```
{
"cmd": "realtime_data",
"devEui": string,
"devAddr": string,
"data": {
"adr": bool,                            //0-Device ADR disabled, 1-Device ADR enabled.
"fcnt": unit32,                         //
"ack": bool,                            //0-ADR ack not required, 1-ADR ack required
"time": string,
"datr": uint8,                          //[0-7] DR0-DR7
"mtype": uint8,
"fopts": string,
"lsnr": float,                          //SNR
"gatewayId": uint32,
"rssi": int8,                           //RSSI
"freq": float,                          //Frequency
"data": string,                         //Hex data
"id": uint64,
"size": int16
}                                       //Raw packet size
}
```

Example:


```
{
    "cmd": "realtime_data",
    "devEui": "7950D20CBBB77561",
    "devAddr": "7950D20C",
    "status" : 1,
    "data": {
                "adr": true,
                "fcnt": 7525,
               "ack": false,
               "time": "2018-02-17 13:06:22+0000",
                "datr": 4,
               "mtype": 2,
                "fopts": "",
               "lsnr": 6.2,
                "gatewayId": "0000B827EBFD64DE",
                "gateway_available" : []
               "rssi": -100,
                "freq": 865.1,
                "data": "7713F800020F010AA78D0014360100150B0000",
               "id": 4671384,
                "size": 32
        }
}
```

3.12. Unsubscribe to realtime data

Request:

```
{
    "cmd": "unconsume_req",
    "unbind": bool                    //Optional
}
```

Accept:

```
{
    "cmd": "unconsume_resp",
    "status": uint8,                  // 1 – OK; 2 - ERROR
    "error_code": uint8               // 1 - Not consuming current APP
}
```

3.13. Checking the integrity of the connection

Request:

```
{
    "cmd": "ping_req"
}
```

Accept:

```
{
    "cmd": "ping_resp",
    "seq": uint16                    // Numerical sequence: 0 - 65353
}
```

3.14. Change timeout of checking the connection integrity

Request:

```
{
cmd": "set_alive_timeout",
"timeout": uint16
}
```

Accept:

```
{
"cmd": "set_alive_timeout_resp",
"timeout": uint16                    // Numerical sequence: 0 - 65353
}
```

3.15. Select data from database

Request:

```
{
    "cmd": "get_data_req",
"operation_id": string,                 //Optional
    "devEui": string,                   // Optional
    "devAddr": string,                  // Optional
    "appId": string,                    // Optional
    "date_from": string,                //Date: "YYYY-MM-DD HH:MM:SS+HHMM"
    "date_to": string,                  // Date: "YYYY-MM-DD HH:MM:SS+HHMM"
    "limit": uint32,                    // Optional – default: 10
    "descending": bool                  // Optional – default: false (if true – descending sort)
}
```

Example request:

```
 {
    "cmd": "get_data_req",
"operation_id": "abc",
    "appId": "77179F01",
    "devEui": "7950D20CBBB77561",
    "date_from": "2018-02-16
00:00:00+0600",
    "date_to": "2018-02-17 00:00:00+0600",
    "limit": 10
}
```

Accept:

```
{
    "cmd": "get_data_resp",
"operation_id": string,
    "status": uint8,                    //1 – Accepted; 2 – ERROR; 3 - SUCCESS
    "error_code": uint8,                //1 - Date parse error,
                                        //2 - Unauthorized request,
                                        //3 - Invalid date range
    "dataCount" : uint32,
    "dataList": [ {
                    "devEui": string,
                  "devAddr": string,
                    "gatewayEui": string,
                  "adr": bool,
                    "fcnt": unit32,
                  "ack": bool,
                   "time": string,
                    "datr": uint8,
```

```
                    "mtype": uint8,
                     "fopts": string,
                    "lsnr": float,
                   "gatewayId": uint32,
                   "rssi": int8,
                   "freq": float,
                   "data": string,
                   "id": uint64,
                    "size": int16,
                    "output_ack": bool,
                    "output_ack_time":
string
                } ]
}
```

Sample accept:

```
{
"cmd": "get_data_resp",
"operation_id": “abc”,
"status" : 1,
"dataCount" : 1,
"dataList": [ {
"devEui": "7950D20CBBB77561",
"devAddr": "7950D20C",
"gatewayEui": "0000B827EB76A75F",
"adr": true,
"fcnt": 7525,
"ack": false,
"time": "2018-02-17 13:06:22+0000",
"datr": 4,
"mtype": 2,
"fopts": "",
"lsnr": 6.2,
"gatewayId": uint32,
"gateway_available" : []
"rssi": -100,
"freq": 865.1,
"data": "7713F800020F010AA78D0014360100150B0000",
"id": 4671384,
"size": 32,
"output_ack": false,
"output_ack_time": null
} ]
}
```

3.16. Downlink - get LoRA WAN Device Class

Request:

```
{
"cmd": "dev_lw_class_req",
"operation_id": string,                    //Optional
"devEui": string                           // Optional
}
```

Example request:

```
{
    "cmd": "dev_lw_class_req",
"operation_id": “axd”,
    "devEui": "6F1E570FFE14DBFC"
}
```

Accept:

```
{
"cmd": "dev_lw_class_resp",
"operation_id": string,
"deviceCount": uint32,
"deviceList": [
{
"devEui": string,
"devAddr": string,
"lwClass": uint8                           // 1 – A; 2 – B; 3 - C
}
]
"status": uint8,                           // 1 – OK; 2 - ERROR
"error_code": uint8                        // 1 - Unauthorized access
                                           // 2 - Required field is not filled
                                           // 3 - Data is filled in with an error
}
```

Sample accept:

{
"cmd": "dev_lw_class_resp",
"operation_id": "axd",
"status": 1,
"deviceCount": 1,
"deviceList": [
{
"devEui": "6F1E570FFE14DBFC",
"devAddr": "78130AE1",
"lwClass": 1
}
]
}

3.17. Downlink - get the maximum data size at the time of the last Uplink

Request:

```
{
    "cmd": "max_dl_data_req",
"operation_id": "abc",                  //Optional
"devEui": string,                       // Optional
    "devAddr": string,                  // Optional
}
```

Example request:

```
{
"cmd": "max_dl_data_req",
"operation_id": "abc",
"devEui": "7AA8974136303736"
}
```

Accept:

```
{
"cmd": "max_dl_data_resp",
"operation_id": string,
"status": uint8,                        // 1 – OK; 2 - ERROR
"error_code": uint8,                    // 1 - Device not found
"devCount": uint8,                      // 2 - Unauthorized access
"devList": [
{
"devEui": string,
"devAddr": string,
"maxDataSize": uint16
}
]
}
```

Sample accept:

{
"cmd": "max_dl_data_resp",
"operation_id": "abc",
"status": 1,
"devCount": 1,
"devList": [
{
"devEui": "7AA8974136303736",
"devAddr": "78130AE1",
"maxDataSize": 51,
}
]
}

3.18. Downlink – send data to device

Request:

```
{
"cmd": "downlink_data_req",
"operation_id": string,                 // Optional
"devEui": string,                       // Optional
"gatewayEui": string,                   // Optional
"freq": float,                          // Optional, MHz
"dr": string,                           // Optional,  (FSK, SF7BW250, SF7BW125, SF8BW125,
"devAddr": string,                      SF9BW125, SF10BW125, SF11BW125, SF12BW125)
    "realTime": bool                    // Optional
"lwClass": uint8,                       // Optional
"fport": uint8,                         // Optional
"data": string
}
```

Example request:

```
{
"cmd": "downlink_data_req",
"operation_id": “xyz”,
"devEui": "6F1E570FFE14DBFC",
"lwClass": 1,
“fport”: 2,
"data": "FFFFFFFFFFFF"
}
```

Accept:

```
{
    "cmd": "downlink_data_resp",
   "operation_id": string,
    "devEui": string,
    "devAddr": string,
    "status": uint8,                    // 1 - OK,  2 - ERROR
    "error_code": uint8,                // 1 - Device not found
    "dataId": uint8,                    // 2 - Unauthorized access
    "dataInQueue": uint8                // 3 - Not HEX in data
}                                       // 4 - Payload max size exceed for current SF
                                        // 5 - Impossible datarate value
                                        // 6 - Frequency must be float
                                        // 7 - Frequency not not in band range
```

Sample accept:

{
"cmd": "downlink_data_resp",
"operation_id": "abc",
"status": 1,
"dataId": 21,
"devAddr": "D1FD37F0",
"devEui": "D745F677B2412E03",
"dataInQueue": 1,
}

Note:
In class A, if a command is delivered to the server in response to an Uplink packet before the timeout for closing the RX1 window expires, the packet will be sent to the gateway, otherwise the packet will be sent in response to the next Uplink packet. If the realTime flag is specified and the delivery is unsuccessful before the timeout for closing the RX1 window expires, the DL packet will be marked as Late and the sending will be canceled.

3.19. Downlink - confirmation of sending to device

Request:

```
{
    "cmd": "downlink_ack_req",
    “operation_id”: string,,                                    //Optional
    "dataId": uint32
}
```

Example request:

```
{
   "cmd": "downlink_ack_req",
   "operation_id": "abc",
   "dataId": 16
}
```

Accept:

```
{
"cmd": "downlink_ack_resp",
“operation_id”: string,
"status": uint8,                    // 1 – OK; 2 - ERROR
"error_code": uint8,                // 1 - dataId is required
                                    // 2 - Device not found
"dataId": uint8,
"ack_time": string
}
```

Sample accept:

```
{
   "cmd": "downlink_ack_resp",
   "operation_id": "abc",
   "ack_time": "2018-06-04 16:30:15+0600",
   "dataId": 16
}
```

3.20. Get sent data from database

Request:

```
{
"cmd": "get_downlink_data_req",
"operation_id": string,                 //Optional
    "devEui": string,                   // Optional
    "devAddr": string,                  // Optional
    "appId": string,                    // Optional
    "date_from": string,                //Data: "YYYY-MM-DD HH:MM:SS+HHMM"
    "date_to": string,                  // Data : "YYYY-MM-DD HH:MM:SS+HHMM"
    "limit": uint32,                    // Optional - default 10
    "descending": bool                  // Optional - default false (true – descending sort)
}
```

Example request:

```
{
"cmd": "get_downlink_data_req",
"operation_id": "rsm234",
"appId": "77179F01",
"devEui": "7950D20CBBB77561",
"date_from": "2019-02-22 00:00:00+0600",
"date_to": "2019-02-23 00:00:00+0600",
"limit": 10
}
```

Accept:

```
{
    "cmd": "get_downlink_data_resp",
"operation_id": string,
    "status": uint8,                    //1 – Accepted; 2 – ERROR; 3 - SUCCESS
    "error_code": uint8,                //1 - Date parse error,
                                        //2 - Unauthorized request,
                                        //3 - Invalid date range
"dataCount" : uint32,
"dataList": [ {
"devEui": string,
"devAddr": string,
"time": string,
"data": string,
"downlink_ack": bool,
"downlink_ack_time": string
} ]
}
```

Sample accept:

{
"cmd": "get_downlink_data_resp",
"operation_id": “rsm234”,
"status" : 1,
"dataCount" : 1,
"dataList": [{
"devEui": "7950D20CBBB77561",
"devAddr": "77179F01",
"time": "2018-02-21 13:06:22+0000",
"data": "7713F800020F010AA78D0014360100150B0000",
"downlink_ack": true,
"downlink_ack_time": "2019-02-22 09:09:00+0600"
}]
}

3.21. Get device coordinates

Request:

```
{
  "cmd": "get_device_coordinates_req",
  "operation_id": string,                    //Optional
  "devEui": string                           // Optional
}
```

Example request:

```
{
  "cmd": "get_device_coordinates_req",
  "operation_id": "1W3fd",
  "devEui": "D745F677B2412E03"
}
```

Accept:

```
{
  "cmd": "get_device_coordinates_resp",
  "operation_id": string,
  "status": uint8,                           // 1 – OK; 2 - ERROR
  "error_code": uint8,                       // 1 - Unauthorized access
                                             // 2 - Required field is not filled
                                             // 3 - Data is filled in with an error
  "deviceCount" : uint32,
  "deviceList": [
    {
      "lat": float,
      "lon": float,
      "devEui": string
    }
  ]
}
```

Sample accept:


```
{
   "cmd": "get_device_coordinates_resp",
   "status": 1,
   "operation_id": "1W3fd",
   "deviceCount": 1,
   "deviceList": [
      {
         "lat": 43.2217921642226,
         "lon": 76.8974626064301,
         "devEui": "D745F677B2412E03"
      }
   ]
}
```

3.22. Set coordinates to device

Request:

```
{
  "cmd": "set_device_coordinates_req",
  "operation_id": string,                    //Optional
  "devEui": string,
  "lat": float,
  "lon": float

}
```

Example request:

```
{
  "cmd": "set_device_coordinates_req",
  "operation_id": "1W3fd",
  "devEui": "D745F677B2412E03",
  "lat": 42.221848,
  "lon": 75.898052

}
```

Accept:

```
{
  "cmd": "set_device_coordinates_resp",
  "operation_id": string,
  "status": uint8,                           // 1 – OK; 2 - ERROR
  "error_code": uint8,                       // 1 - Unauthorized access
                                             // 2 - Required field is not filled
                                             // 3 - Data is filled in with an error
                                             // 8 - Device does not exists
  "devEui": string,
  "lat": float,
  "lon": float
}
```

Sample accept:

```
{
  "cmd": "set_device_coordinates_resp",
  "status": 1,
  "operation_id": "1W3fd",
  "devEui": "D745F677B2412E03",
  "lat": 42.221848,
  "lon": 75.898052
}
```

3.23. Additional commands

Accept:

```
{
    "cmd": "json_parse_error"
}
```

Accept:

```
{
    "cmd": "invalid_req"
}
```

Accept:

```
{
    "cmd": "invalid_command_req"
}
```

Accept:

```
{
    "cmd": "die_connection_close"
}
```

4. REVISION SHEET

Version	Date	Name	Comments
01	11.06.2017	DF	Document creation date
02	22.05.2018	DF	Create/Update/Delete app (BETA) Create/Update/Delete devices (BETA)
03	07.06.2018	RS	Create/Update/Delete app (UPDATE) Create/Update/Delete devices (UPDATE)
04	20.06.2018	DF	Correction in text
05	01.08.2018	DF	Clarification (0-OTAA, 1-ABP)
06	06.09.2018	DF	Clarification the Realtime data
07	12.12.2018	RS	Create/Update/Delete tariffs (UPDATE)
08	22.02.2019	RS	Get sent data from database
09	09.04.2019	RS	Added the Get device coordinates method Added the Set device coordinates method Coordinate fields added (lat, lon) in methods: Create device, Update devices, Get device Removed devAddr parameter from methods: Get devices, Get LoRA WAN device class
10	29.07.2019	DF	Adding die_connection_close.
11	20.04.2020	DF	Added the "realTime": bool in Downlink – send dada to device
12	28.12.2020	DF	Translate in English; Added methods connections_list, connection_kill
13	18.01.2021r.	DF	Added methods set_alive_timeout. Updated methods downlink_data_req. Correction in text.