Plat' Home
TECHNOLOGY to serve you.

# OpenBlocks IoT Family PD Handler JSON Format List


**Ver.3.1.0**

**Plat'Home Co., Ltd.**

## ■ About trademarks

- Company names, product names and other names in this document may be trademarks or registered trademarks of each company.
- Other proper names such as product names in this document are trademarks or registered trademarks of each company.

## ■ Before use

- It is prohibited to reprint the contents of this document in whole or in part without prior consent.
- The contents of this document are subject to change without notice.
- Although we make our best efforts to ensure the accuracy of this document, please contact our support center if you find typographical errors or other mistakes.
  The latest version of this document can be downloaded from our website.
- Note in advance that this device is not assumed to be used in in fields with fatal dangers.
- Note in advance that we bear no responsibility for any damages and lost earnings resulting from operation of this device.

## ■ Notes

□ This document does not contain information on devices destined for Japan that are not exported abroad.

## Table of Contents

# 1.PD Handler BLE (Node.js)

## 1.1.Beacon

### 1.1.1.When setting to use as a beacon is done

■Data Sample

```
{
        “time”:” 2017-12-08T15:00:04.549+09:00”,
        “deviceId”:” e9c8dd35ee18”,
        “appendixInfo”:” G8H00012”,
        “rssi”:-88,
        “type”:”iBeacon”,
        “data”:” 0201040c0946434c20426561636f6e31”,
        “localname”:”beacon”,
        “status”:”in”
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”"or “out”) |
| ex | User define | User defined | String | | Value set in Web UI. |

### 1.1.2.When setting to use as BLE is done

■Data Sample

```
{
        “time”:” 2017-12-08T15:00:04.549+09:00”,
        “deviceId”:” e9c8dd35ee18”,
        “memo”:”BLE beacon”
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 3 | memo | Notes | String | | Value set in Web UI. |

# 1.2.Sensor

## 1.2.1TI Sensor

■Data sample


```
{
        “deviceId”:”b0b448b93907”,
        “time”:”2016-03-14T09:32:15.864+09:00”,
        “humidity”:68.12,
        “temperature”:25.51,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.1001,
        “gyroX”:0.3002,
        “gyroY”:0.9001,
        “gyroZ”:2.1003,
        “magX”:-25.5004,
        “magY”:48.0001,
        “magZ”:-159.2002,
        “pressure”:1008.22,
        “objectTemp”:21,
        “ambientTemp”:25.3,
        “lux”:0.2
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | humidity | Humidity | Double | △ | [%] |
| 4 | temperature | Temperature | Double | △ | [℃] |
| 5 | accelX | X axis acceleration | Double | △ | [G] |
| 6 | accelY | Y axis acceleration | Double | △ | [G] |
| 7 | accelZ | Z axis acceleration | Double | △ | [G] |
| 8 | gyroX | X axis angular velocity | Double | △ | [°/s] |
| 9 | gyroY | Y axis angular velocity | Double | △ | [°/s] |
| 10 | gyroZ | Z axis angular velocity | Double | △ | [°/s] |
| 11 | magX | X axis geomagnetic torque | Double | △ | [μT] |
| 12 | magY | Y axis geomagnetic torque | Double | △ | [μT] |
| 13 | magZ | Z axis geomagnetic torque | Double | △ | [μT] |
| 14 | pressure | Atmospheric pressure | Double | △ | [hPa] |
| 15 | objectTemp | Object temperature | Double | △ | [℃] |
| 16 | ambientTemp | Ambient temperature | Double | △ | [℃] |
| 17 | lux | Illuminance | Double | △ | [lux] |
| 18 | memo | Notes | String | | Value set in Web UI. |

*Depending on the remaining battery level and sensor model, there is data not included

## 1.2.1.Fujitsu Sensor

### 1.2.1.1.When setting to use as a beacon is done

■Data sample


```
{
        “deviceId”:”b0b448b93908”,
        “time”:”2016-03-14T09:12:15.225+09:00”,
        “rssi”:-67,
        “temperature”:25.61,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0001
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”’or “out”) |
| 9 | temperature | Temperature | Double | △ | [℃] |
| 10 | accelX | X axis acceleration | Double | △ | [G] |
| 11 | accelY | Y axis acceleration | Double | △ | [G] |
| 12 | accelZ | Z axis acceleration | Double | △ | [G] |
| ex | User define | User defined | String | | Value set in Web UI. |

## 1.2.1.2.When setting to use as BLE is done

■Data sample


```
{
        “deviceId”:”b0b448b93908”,
        “time”:”2016-03-14T09:12:15.225+09:00”,
        “temperature”:25.61,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0001
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | temperature | Temperature | Double | ○ | [℃] |
| 4 | accelX | X axis acceleration | Double | ○ | [G] |
| 5 | accelY | Y axis acceleration | Double | ○ | [G] |
| 6 | accelZ | Z axis acceleration | Double | ○ | [G] |
| 7 | memo | Notes | String | | Value set in Web UI. |

# 1.2.3.ALPS　IoT Smart Module

## 1.2.3.1.When setting to use as a beacon is done

■Data sample(Beacon mode: Environmental format)


```
{
        “time”:”2016-03-14T17:05:42.965+09:00”,
        “deviceId”:”34c731ffe620”,
        “rssi”:-87,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0002,
        “pressure”:1010.42,
        “humidity”:58.83,
        “temperature”:29.41,
        “uv”:0.0515,
        “ambientLight”:50.5368
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”or “out”) |
| 9 | accelX | X axis acceleration | Double | | [G] |
| 10 | accelY | Y axis acceleration | Double | | [G] |
| 11 | accelZ | Z axis acceleration | Double | | [G] |
| 12 | pressure | Atmospheric pressure | Double | | [hPa] |
| 13 | humidity | Humidity | Double | | [%] |
| 14 | temperature | Temperature | Double | | [℃] |
| 15 | uv | Ultraviolet | Double | | [mW/cm2] |
| 16 | ambientLight | Illuminance | Double | | [lux] |
| ex | User define | User defined | String | | Value set in Web UI. |

■Data sample(Beacon mode: Motion format)


```
{
        “time”:”2016-03-14T17:05:42.965+09:00”,
        “deviceId”:”34c731ffe620”,
        “rssi”:-87,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0,
        “geoMagneticX”:25.35,
        “geoMagneticY”:-35.70,
        “geoMagneticZ”:7.05,
        “pressure”:1010.42
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”’or “out”) |
| 9 | accelX | X axis acceleration | Double | | [G] |
| 10 | accelY | Y axis acceleration | Double | | [G] |
| 11 | accelZ | Z axis acceleration | Double | | [G] |
| 12 | geoMagneticX | X axis geomagnetic torque | Double | | [uT] |
| 13 | geoMagneticY | Y axis geomagnetic torque | Double | | [uT] |
| 14 | geoMagneticZ | Z axis geomagnetic torque | Double | | [uT] |
| 15 | pressure | Atmospheric pressure | Double | | [hPa] |
| ex | User define | User defined | String | | Value set in Web UI. |

## 1.2.3.2.When setting to use as BLE is done

■Data sample(Connection mode: Data packet 1)


```
{
        “deviceId”:”34c731ffe620”,
        “time”:”2016-07-14T09:12:29.231+09:00”,
        “dataIndex”:123,
        “geoMagneticX”:25.35,
        “geoMagneticY”:-35.70,
        “geoMagneticZ”:7.05,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0001,
        “ms”:0,
        “second”:28,
        “minute”:12,
        “hour”:9
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | dataIndex | Data index | Integer | ○ | 0~255 (Sequence number) |
| 4 | geoMagneticX | X axis geomagnetic torque | Double | | [uT] |
| 5 | geoMagneticY | Y axis geomagnetic torque | Double | | [uT] |
| 6 | geoMagneticZ | Z axis geomagnetic torque | Double | | [uT] |
| 7 | accelX | X axis acceleration | Double | | [G] |
| 8 | accelY | Y axis acceleration | Double | | [G] |
| 9 | accelZ | Z axis acceleration | Double | | [G] |
| 10 | ms | Millisecond | Integer | ○ | |
| 11 | second | Second | Integer | ○ | |
| 12 | minute | Minute | Integer | ○ | |
| 13 | hour | Hour | Integer | ○ | |
| 14 | memo | Notes | String | | Value set in Web UI. |

■Data sample(Connection mode: Data packet 2)


```
{
        “deviceId”:”34c731ffe620”,
        “time”:”2016-07-14T09:12:29.456+09:00”,
        “dataIndex”:123,
        “pressure”:1010.42,
        “humidity”:58.83,
        “temperature”:29.41,
        “uv”:0.0515,
        “ambientLight”:50.5368,
        “day”:14,
        “month”:7,
        “year”:16
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | dataIndex | Data index | Integer | ○ | 0~255 (Sequence number) |
| 4 | pressure | Atmospheric pressure | Double | | [hPa] |
| 5 | humidity | Humidity | Double | | [%] |
| 6 | temperature | Temperature | Double | | [℃] |
| 7 | uv | Ultraviolet | Double | | [mW/cm2] |
| 8 | ambientLight | Illuminance | Double | | [lux] |
| 9 | day | Day | Integer | ○ | |
| 10 | month | Month | Integer | ○ | |
| 11 | year | Year | Integer | ○ | |
| 12 | memo | Notes | String | | Value set in Web UI. |

■Data sample (beacon mode: environmental format)


```
{
        “time”:”2016-03-14T17:05:42.965+09:00”,
        “memo”:”ALPS beacon env”,
        “deviceId”:”34c731ffe620”,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0002,
        “pressure”:1010.42,
        “humidity”:58.83,
        “temperature”:29.41,
        “uv”:0.0515,
        “ambientLight”:50.5368
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | accelX | X axis acceleration | Double | | [G] |
| 4 | accelY | Y axis acceleration | Double | | [G] |
| 5 | accelZ | Z axis acceleration | Double | | [G] |
| 6 | pressure | Atmospheric pressure | Double | | [hPa] |
| 7 | humidity | Humidity | Double | | [%] |
| 8 | temperature | Temperature | Double | | [℃] |
| 9 | uv | Ultraviolet | Double | | [mW/cm2] |
| 10 | ambientLight | Illuminance | Double | | [lux] |
| 11 | memo | Notes | String | | Value set in Web UI. |

■Data sample (beacon mode: motion format)

```
{
        “time”:”2016-03-14T17:05:42.965+09:00”,
        “deviceId”:”34c731ffe620”,
        “memo”:”ALPS beacon motion”,
        “accelX”:0,
        “accelY”:0,
        “accelZ”:-1.0,
        “geoMagneticX”:25.35,
        “geoMagneticY”:-35.70,
        “geoMagneticZ”:7.05,
        “pressure”:1010.42
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | accelX | X axis acceleration | Double | | [G] |
| 4 | accelY | Y axis acceleration | Double | | [G] |
| 5 | accelZ | Z axis acceleration | Double | | [G] |
| 6 | geoMagneticX | X axis geomagnetic torque | Double | | [uT] |
| 7 | geoMagneticY | Y axis geomagnetic torque | Double | | [uT] |
| 8 | geoMagneticZ | Z axis geomagnetic torque | Double | | [uT] |
| 9 | pressure | Atmospheric pressure | Double | | [hPa] |
| 10 | memo | Notes | String | | Value set in Web UI. |

# 1.2.4.OMRON　Environmental Sensor

## 1.2.4.1.When setting to use as a beacon is done

■Data sample (beacon mode: IM)


```
{
        “time”:”2016-10-14T18:23:27.739+09:00”,
        “deviceId”:”d11397e0d126”,
        “rssi”:-61,
        “sequence”:36349,
        “temperature”:24.39,
        “humidity”:39.23,
        “light”:93,
        “uvi”:0.18,
        “pressure”:1013.5,
        “noise”:39.26,
        “accelX”:-0.3,
        “accelY”:0.1,
        “accelZ”:1.2,
        “battery”:2930
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”’or “out”) |
| 9 | sequence | Sequence number | Integer | ○ | |
| 10 | temperature | Temperature | Double | ○ | [℃] |
| 11 | humidity | Humidity | Double | ○ | [%] |
| 12 | light | Illuminance | Integer | ○ | [lux] |
| 13 | uvi | UV index | Double | ○ | |
| 14 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 15 | noise | Noise | Double | ○ | [dB] |
| 16 | accelX | X axis acceleration | Double | | [G] |
| 17 | accelY | Y axis acceleration | Double | | [G] |
| 18 | accelZ | Z axis acceleration | Double | | [G] |
| 19 | battery | Battery voltage | Integer | ○ | [mV] |
| ex | User define | User defined | String | | Value set in Web UI. |

■Data sample (beacon mode: EP)


```
{
        “time”:”2016-10-14T18:05:22.375+09:00”,
        “deviceId”:”d11397e0d126”,
        “rssi”:-61,
        “sequence”:36381,
        “temperature”:24.46,
        “humidity”:39.73,
        “light”:97,
        “uvi”:0.03,
        “pressure”:1013.2,
        “noise”:39.42,
        “discomfortIndex”:70.33,
        “heatstroke”:19.77,
        “battery”:2910
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”’or “out”) |
| 9 | sequence | Sequence number | Integer | ○ | |
| 10 | temperature | Temperature | Double | ○ | [℃] |
| 11 | humidity | Humidity | Double | ○ | [%] |
| 12 | light | Illuminance | Integer | ○ | [lux] |
| 13 | uvi | UV index | Double | ○ | |
| 14 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 15 | noise | Noise | Double | ○ | [dB] |
| 16 | discomfortIndex | Discomfort index | Double | ○ | |
| 17 | heatstroke | Heat stroke risk | Double | ○ | [℃] |
| 18 | battery | Battery voltage | Integer | ○ | [mV] |
| ex | User define | User defined | String | | Value set in Web UI. |

## 1.2.4.2.When setting to use as BLE is done

■Data sample (connection mode)


```
{
        “deviceId”:”d11397e0d126”,
        “memo”:”OMRON Env Sensor”
        “time”:”2016-10-14T09:27:52.278+09:00”,
        “humidity”:38.7,
        “temperature”:25.42,
        “light”:114,
        “uvi”:0.02,
        “pressure”:1018.1,
        “noise”:38.17,
        “discomfortIndex”:71.09,
        “heatstroke”:20.05,
        “battery”:2917
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | humidity | Humidity | Double | ○ | [%] |
| 4 | temperature | Temperature | Double | ○ | [℃] |
| 5 | light | Illuminance | Integer | ○ | [lux] |
| 6 | uvi | UV index | Double | ○ | |
| 7 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 8 | noise | Noise | Double | ○ | [dB] |
| 9 | discomfortIndex | Discomfort index | Double | ○ | |
| 10 | heatstroke | Heat stroke risk | Double | ○ | [℃] |
| 11 | battery | Battery voltage | Integer | ○ | [mV] |
| 12 | memo | Notes | String | | Value set in Web UI. |

■Data sample (beacon mode: IM)


```
{
        “time”:”2016-10-14T18:23:27.739+09:00”,
        “memo”:”OMRON Env Sensor IM”
        “deviceId”:”d11397e0d126”,
        “sequence”:36349,
        “temperature”:24.39,
        “humidity”:39.23,
        “light”:93,
        “uvi”:0.18,
        “pressure”:1013.5,
        “noise”:39.26,
        “accelX”:-0.3,
        “accelY”:0.1,
        “accelZ”:1.2,
        “battery”:2930
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | sequence | Sequence number | Integer | ○ | |
| 4 | temperature | Temperature | Double | ○ | [℃] |
| 5 | humidity | Humidity | Double | ○ | [%] |
| 6 | light | Illuminance | Integer | ○ | [lux] |
| 7 | uvi | UV index | Double | ○ | |
| 8 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 9 | noise | Noise | Double | ○ | [dB] |
| 10 | accelX | X axis acceleration | Double | | [G] |
| 11 | accelY | Y axis acceleration | Double | | [G] |
| 12 | accelZ | Z axis acceleration | Double | | [G] |
| 13 | battery | Battery voltage | Integer | ○ | [mV] |
| 14 | memo | Notes | String | | Value set in Web UI. |

■Data sample (beacon mode: EP)


```
{
        “time”:”2016-10-14T18:05:22.375+09:00”,
        “memo”:”OMRON Env Sensor EP”
        “deviceId”:”d11397e0d126”,
        “sequence”:36381,
        “temperature”:24.46,
        “humidity”:39.73,
        “light”:97,
        “uvi”:0.03,
        “pressure”:1013.2,
        “noise”:39.42,
        “discomfortIndex”:70.33,
        “heatstroke”:19.77,
        “battery”:2910
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | sequence | Sequence number | Integer | ○ | |
| 4 | temperature | Temperature | Double | ○ | [℃] |
| 5 | humidity | Humidity | Double | ○ | [%] |
| 6 | light | Illuminance | Integer | ○ | [lux] |
| 7 | uvi | UV index | Double | ○ | |
| 8 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 9 | noise | Noise | Double | ○ | [dB] |
| 10 | discomfortIndex | Discomfort index | Double | ○ | |
| 11 | heatstroke | Heat stroke risk | Double | ○ | [℃] |
| 12 | battery | Battery voltage | Integer | ○ | [mV] |
| 13 | memo | Notes | String | | Value set in Web UI. |

## 1.2.5.UNI-ELECTRONICS Wireless thermohygrometer

### 1.2.5.1.When setting to use as a beacon is done

■Data sample (beacon mode)

```
{
        “time”:”2016-10-14T11:30:41.259+09:00”,
        “deviceId”:”f0ab542bdca5”,
        “rssi”:-90,
        “temperature”:27.88,
        “humidity”:36.48,
        “battery”:100
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”“or “out”) |
| 9 | temperature | Temperature | Double | ○ | [℃] |
| 10 | humidity | Humidity | Double | ○ | [%] |
| 11 | battery | Battery usage rate | Integer | ○ | [%] |
| ex | User define | User defined | String | | Value set in Web UI. |

### 1.2.5.2.When setting to use as BLE is done

■Data sample (beacon mode)

```
{
        “time”:”2016-10-14T11:30:41.259+09:00”,
        “deviceId”:”f0ab542bdca5”,
        “memo”:”Logtta TH Sensor”,
        “temperature”:27.88,
        “humidity”:36.48,
        “battery”:100
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | temperature | Temperature | Double | ○ | [℃] |
| 4 | humidity | Humidity | Double | ○ | [%] |
| 5 | battery | Battery usage rate | Integer | ○ | [%] |
| 6 | memo | Notes | String | | Value set in Web UI. |

# 1.2.6.UNI-ELECTRONICS Wireless Carbon Dioxide Sensor

## 1.2.6.1.When setting to use as a beacon is done

■Data sample (beacon mode)

```
{
        “time”:”2017-03-03T12:34:56.789+09:00”,
        “deviceId”:”f0ab54c2gcdf”,
        “rssi”:-82,
        “co2”:653,
        “battery”:254
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”“or “out”) |
| 9 | co2 | Carbon dioxide concentration | Integer | ○ | [ppm] |
| 10 | battery | Battery usage rate | Integer | ○ | [%] |
| ex | User define | User defined | String | | Value set in Web UI. |

## 1.2.5.2.When setting to use as BLE is done

■Data sample (beacon mode)

```
{
        “time”:”2017-03-03T12:34:56.789+09:00”,
        “deviceId”:”f0ab54c2gcdf”,
        “memo”:”Logtta CO2 Sensor”,
        “co2”:653,
        “battery”:254
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | co2 | Carbon dioxide concentration | Integer | ○ | [ppm] |
| 4 | battery | Battery usage rate | Integer | ○ | [%] |
| 5 | memo | Notes | String | | Value set in Web UI. |

# 1.2.7.UNI-ELECTRONICS Wireless Water Thermometer

## 1.2.7.1.When setting to use as a beacon is done

■Data sample (beacon mode)


```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:”f0ab5e2bdcad”,
        “rssi”:-82,
        “temperature”:12.34,
        “battery”:100
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 8 | status | Beacon status | String | | Displayed by beacon control type. (“In”“or “out”) |
| 3 | temperature | Temperature | Double | ○ | [℃] |
| 4 | battery | Battery usage rate | Integer | ○ | [%] |
| ex | User define | User defined | String | | Value set in Web UI. |

## 1.2.7.2.When setting to use as BLE is done

■Data sample (beacon mode)


```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:”f0ab5e2bdcad”,
        “memo”:”Logtta Water Sensor”,
        “temperature”:12.34,
        “battery”:100
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | temperature | Temperature | Double | ○ | [℃] |
| 4 | battery | Battery usage rate | Integer | ○ | [%] |
| 5 | memo | Notes | String | | Value set in Web UI. |

## 1.2.8.RATOC systems　Bluetooth Dust Sensor

■Data sample


```
{
        “deviceId”:”dfb3f8c57912”,
        “memo”:”RATOC PM2.5”,
        “time”:”2017-12-07T20:55:48.173+09:00”,
        "sensortime":"17-12-07T20:56:04",
        “pm25”:15,
        “pm10”1,
        “pressure”:999,
        “temperature”:24,
        “humidity”:18,
        “light”:364,
        “mode”:0
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | sensortime | Time measured | String | ○ | |
| 4 | pm25 | PM2.5 concentration | Integer | ○ | [μg/m3] |
| 5 | pm10 | PM10 concentration | Integer | ○ | [μg/m3] |
| 6 | pressure | Atmospheric pressure | Integer | ○ | [hPa] |
| 7 | temperature | Temperature | Integer | ○ | [℃] |
| 8 | humidity | Humidity | Integer | ○ | [%] |
| 9 | light | Illuminance | Integer | ○ | [lx] |
| 10 | mode | Measurement mode | Integer | ○ | 0: Continuous, 1: One shot |
| 11 | memo | Notes | String | | Value set in Web UI. |

# 1.2.9.RATOC systems　Bluetooth Air Quality Monitor

■Data sample


```
{
        “deviceId”:”dfb308abcdef”,
        “memo”:”RATOC PM2.5V”,
        “time”:”2017-12-07T20:55:48.173+09:00”,
        "sensortime":"17-12-07T20:56:04",
        “pm25”:15,
        “pm10”:1,
        “uvi”: 0,
        “temperature”:24,5,
        “humidity”:18,1,
        “phumidity”:18,2,
        “pressure”:999.9,
        “initstate”:” wait”,
        “startstate”:” stability”,
        “light”:364,
        “tvoc”:123,
        “eco2”:456,
        “mode”:0
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | sensortime | Time measured | String | ○ | |
| 4 | pm25 | PM2.5 concentration | Integer | ○ | [μg/m3] |
| 5 | pm10 | PM10 concentration | Integer | ○ | [μg/m3] |
| 6 | uvi | UV index | Integer | ○ | |
| 7 | temperature | Temperature | Double | ○ | [℃] |
| 8 | humidity | Humidity | Double | ○ | [%] |
| 9 | phumidity | Humidity on atmospheric pressure sensor element. | Double | ○ | [%] |
| 10 | pressure | Atmospheric pressure | Double | ○ | [hPa] |
| 11 | initstate | Initial stable state | String | ○ | “wait” or ”stability” |
| 12 | startstate | Stable state at startup. | String | ○ | “wait” or ”stability” |
| 13 | light | Illuminance | Integer | ○ | [lx] |
| 14 | tvoc | Total volatile organic compounds. | Integer | ○ | [ppb] |
| 15 | eco2 | Equivalents carbon dioxide | Integer | ○ | [ppm] |
| 16 | mode | Measurement mode | Integer | ○ | 0: Continuous, 1: One shot |
| 17 | memo | Notes | String | | Value set in Web UI. |

## 1.2.10.RATOC systems　Bluetppth Watt Checker

■Data sample


```
{
        “deviceId”:”123456abcdef”,
        “memo”:”RATOC WATT CHECKER”,
        “time”:”2017-12-07T20:55:48.173+09:00”,
        "sensortime":"17-12-07T20:56:04",
        “current”:17.7656,
        “voltage”:104728,
        “power_consumption”:400
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 2 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 3 | sensortime | Time measured | String | ○ | |
| 4 | current | Current | Double | ○ | [mA] |
| 5 | voltage | Voltage | Integer | ○ | [V] |
| 6 | power_consumption | Power consumption | Integer | ○ | [W] |
| 7 | memo | Notes | String | | Value set in Web UI. |

# 2.PD Handler BLE with Lua on C language

The JSON data output to the log is in no particular order.

# 2.1.Beacon

Show subsection 1.1 “Beacon” of section 1 “PD Handler BLE with Node.js”.

# 2.2.Sensor

It does not support to sensors in connection mode.

Please refer to beacon mode written in sub session 1.2 "Sensor" of session 1 "PD Handler BLE with Node.js " for supported sensor.

In addition, rssi information is also given when setting to use BLE.

Below is a description of devices operating with PD Handler BLE with Lua.

# 2.2.1.NAKAYO　Push Botton

# 2.2.1.1.When setting to use as a beacon is done

■Data sample

```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:” fc97c1aef545”,
        “rssi”:-68
        “uuid”: “a903010014784824b2988e6823cfdefa”,
        “major”:”00c8”,
        “minor”:”ffe0”,
        “push”:0
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 3 | appendixInfo | Appendix Information | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication | Integer | ○ | |
| 5 | type | Beacon type | String | | Value set in Web UI. |
| 6 | data | Body of the data | String | | Hexadecimal data |
| 7 | localname | Local name | String | | Value set in Web UI. |
| 5 | uuid | uuid | String | ○ | |
| 6 | major | major | String | ○ | |
| 7 | minor | minor | String | ○ | |
| 8 | push | Pushed botton | Integer | ○ | Every time the button is pushed, 0 to 3 are output in order. |
| ex | User define | User defined | String | | Value set in Web UI. |

## 2.2.1.2.When setting to use as BLE is done

■Data sample


```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:” fc97c1aef545”,
        “memo”:”Nakayo”,
        “rssi”:-68
        “uuid”: “a9030100l4784824b2988e6823cfdefa”,
        “major”:”00c8”,
        “minor”:”ffe0”,
        “push”:0
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | deviceId | Device ID | String | ○ | Excluding “:” from the device address, lower case value. |
| 3 | memo | Notes | String | | Value set in Web UI. |
| 4 | rssi | Received signal strength indication. | Integer | ○ | |
| 5 | uuid | uuid | String | ○ | |
| 6 | major | major | String | ○ | |
| 7 | minor | minor | String | ○ | |
| 8 | push | Pushed botton | Integer | ○ | Every time the button is pushed, 0 to 3 are output in order. |

# 3.PD Handler Modbus

## 3.1.Modbus Client(Modbus master)

### 3.1.1.Polling operation to PLC device

■Data sample (reading register output or register input by TCP protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “protocol”:”tcp”,
        “node”:”172.16.7.250”,
        “port”:1502,
        “unit”:255,
        “memo”:”PLC01”,
        “address”:31,
        “function”:3,
        “data_type”:”uint16_t”,
        “values”:[2,0,1234,5678,9876]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 3 | node | IP address of PLC device | String | ○ | Value set in Web UI.<br>“tcp”only |
| 4 | port | TCP port number | Integer | ○ | Value set in Web UI.<br>“tcp”only |
| 5 | device | Device file name | String | | Value set in Web UI.<br>“rtu”only |
| 6 | unit | Modbus Unit ID | Integer | ○ | Value set in Web UI. |
| 7 | memo | Notes | String | ○ | Value set in Web UI. |
| 8 | address | Registers address to be read data. | Integer | ○ | Value set in Web UI. |
| 9 | function | Modbus function code | integer | ○ | Value set in Web UI.<br>3: Register output<br>4: Register input |
| 10 | data_type | Data type | String | ○ | Value set in Web UI.<br>“uint16_t” :<br>Unsigned 16bits<br>“int16_t”:<br>Signed16bits<br>“uint32lsb_t”:<br>Unsigned 32bits LSB<br>“uint32msb_t”:<br>Unsigned 32bits MSB<br>“int32lsb_t”:<br>Signed 32bits LSB<br>“int32msb_t”:<br>Signed 32bits MSB |
| 11 | values | Body of the data | Integer array | ○ | The number of arrays is variable according to the setting of the number of read registers. |

■Data sample (reading of digital output or digital input by RTU protocol)


```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “memo”:”PLC04”,
        “address”:37,
        “function”:2,
        “values”:[1,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1,0,0,1,1,1]
}
```


| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 3 | node | IP address of PLC device. | String | ○ | Value set in Web UI.<br>“tcp” only |
| 4 | port | TCP port number | Integer | ○ | Value set in Web UI.<br>“tcp” only |
| 5 | device | Device file name | String | | Value set in Web UI.<br>“rtu” only |
| 6 | unit | Modbus Unit ID | Integer | ○ | Value set in Web UI. |
| 7 | memo | Notes | String | ○ | Value set in Web UI. |
| 8 | address | Registers address from which data was read. | Integer | ○ | Value set in Web UI. |
| 9 | function | Modbus function code | Integer | ○ | Value set in Web UI.<br>1: Digital output<br>2: Digital input |
| 10 | values | Body of the data | Integer array | ○ | 0 or 1.<br>The number of arrays is variable according to the setting of the number of read registers. |

## 3.1.2.On demand operation from the cloud

■Request message sample (reading register output or register input by TCP protocol)

```
{
        “protocol”:”tcp”,
        “node”:”172.16.7.250”,
        “port”:1502,
        “unit”:255,
        “address”:31,
        “function”:3,
        “number”:5,
        “data_type”:”uint16_t”
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | protocol | Protocol | String | ○ | “tcp” or ”rtu” |
| 2 | node | IP address of PLC device | String | △ | IP address of PLC device, required for “tcp”. |
| 3 | port | TCP port number | Integer | △ | TCP port number of PLC device, required for “tcp”. |
| 4 | device | Device file name | String | △ | Serial port connecting PLC equipment, required for “rtu”. |
| 5 | unit | Modbus Unit ID | Integer | △ | Must be for “rtu”.<br>If omitted at "tcp", 255. |
| 6 | address | Registers address to be read data. | Integer *1 | | If omitted, 0. |
| 7 | function | Modbus function code | Integer *1 | ○ | 3: read holding registers<br>4: read input registers |
| 8 | number | Number of register to be read. | Integer *1 | | If omitted, 1. |
| 9 | data_type | Data type | String | | “uint16_t” :<br>Unsigned 16bits<br>“int16_t”:<br>Signed16bits<br>“uint32lsb_t”:<br>Unsigned 32bits LSB<br>“uint32msb_t”:<br>Unsigned 32bits MSB<br>“int32lsb_t”:<br>Signed 32bits LSB<br>“int32msb_t”:<br>Signed 32bits MSB<br>If omitted,”uint16_t”. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (Register output by TCP protocol or reading register input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”84bfb66e5a0841732e28463bb91c297c”,
        “result”:”done”,
        “protocol”:”tcp”,
        “node”:”172.16.7.250”,
        “port”:1502,
        “unit”:255,
        “memo”:”PLC01”,
        “address”:31,
        “function”:3,
        “data_type”:”uint16_t”,
        “values”:[2,0,1234,5678,9876]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Requested value.<br>“tcp” or ”rtu” |
| 5 | node | IP address of PLC device | String | | Requested value.<br>“tcp” only |
| 6 | port | TCP port number | Integer | | Requested value.<br>“tcp” only |
| 7 | device | Device file name | String | | Requested value.<br>“rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | Requested value. |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was read. | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | integer | ○ | Requested value. |
| 12 | data_type | Data type | String | ○ | Requested value. |
| 13 | values | Body of the data | Integer array | ○ | The number of arrays is variable according to the number of registers requested with the number key. |

■ Request message sample (reading of digital output or digital input by RTU protocol)

```
{
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485
        “unit”:21,
        “address”:37,
        “function”:2,
        “number”:20
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | protocol | Protocol | String | ○ | “tcp” or ”rtu” |
| 2 | node | IP address of PLC device | String | △ | IP address of PLC device, required for “tcp”. |
| 3 | port | TCP port number | Integer | △ | TCP port number of PLC device, required for “tcp”. |
| 4 | device | Device file name | String | △ | Serial port connecting PLC equipment, required for “rtu”. |
| 5 | unit | Modbus Unit ID | Integer | △ | Must be for “rtu”.<br>If omitted at "tcp", 255. |
| 6 | address | Registers address to be read data. | Integer *1 | | If omitted, 0. |
| 7 | function | Modbus function code | Integer *1 | ○ | 1: read coils<br>2: read discrete inputs |
| 8 | number | Number of bit to be read. | Integer *1 | | If omitted, 1. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (reading of digital output or digital input by RTU protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”5762a76a3235c71c5759029f078a8ca2”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “memo”:”PLC04”,
        “address”:37,
        “function”:2,
        “values”:[1,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1,0,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Requested value.<br>“tcp” or ”rtu” |
| 5 | node | IP address of PLC device | String | | Requested value.<br>“tcp” only |
| 6 | port | TCP port number | Integer | | Requested value.<br>“tcp” only |
| 7 | device | Device file name | String | | Requested value.<br>“rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | Requested value. |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was read. | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | integer | ○ | Requested value. |
| 12 | values | Body of the data | Integer array | ○ | 0 or 1.<br>The number of arrays is variable according to the number of registers requested with the number key. |

■Request message sample (write to register input by TCP protocol)

```
{
        “protocol”:”tcp”,
        “node”:”172.16.7.250”,
        “port”:1502,
        “unit”:255,
        “address”:”0x0ab”,
        “function”:16,
        “data_type”:”uint32lsb_t”,
        “values”:[42949672951,21474836471]
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | protocol | Protocol | String | ○ | “tcp” or ”rtu” |
| 2 | node | IP address of PLC device | String | △ | IP address of PLC device, required for “tcp”. |
| 3 | port | TCP port number | Integer | △ | TCP port number of PLC device, required for “tcp”. |
| 4 | device | Device file name | String | △ | Serial port connecting PLC equipment, required for “rtu”. |
| 5 | unit | Modbus Unit ID | Integer | △ | Must be for “rtu”. If omitted at "tcp", 255. |
| 6 | address | Registers address to be write data. | Integer *1 | | If omitted, 0. |
| 7 | function | Modbus function code | Integer *1 | ○ | 6:write_single_register 16:write_multiple_registers 23:write_and_read_registers |
| 8 | data_type | Data type | String | | “uint16_t” : Unsigned 16bits “int16_t”: Signed16bits “uint32lsb_t”: Unsigned 32bits LSB “uint32msb_t”: Unsigned 32bits MSB “int32lsb_t”: Signed 32bits LSB “int32msb_t”: Signed 32bits MSB If omitted,”uint16_t” |
| 9 | values | Body of the write data | Integer array | ○ | If the function key is 6, write the first 1 register. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (write to register input by TCP protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”73771103b4765ed0ce859ac912321c04”,
        “result”:”done”
        “protocol”:”tcp”,
        “node”:”172.16.7.250”,
        “port”:1502,
        “unit”:255,
        “address”:”0x0ab”,
        “function”:16,
        “data_type”:”uint32lsb_t”,
        “values”:[42949672951,21474836471]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Requested value.<br>“tcp” or ”rtu” |
| 5 | node | IP address of PLC device | String | | Requested value.<br>“tcp” only |
| 6 | port | TCP port number | Integer | | Requested value.<br>“tcp” only |
| 7 | device | Device file name | String | | Requested value.<br>“rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | Requested value. |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was wrtten | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | Integer | ○ | Requested value. |
| 12 | data_type | Data type | String | ○ | Requested value. |
| 13 | values | Body of the written data. | Integer array | ○ | Requested value. |

■Request message sample (writing to digital input by RTU protocol)

```
{
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “address”:”0x0ce”,
        “function”:15,
        “values”:[0,0,0,1,1,1,0,1,1,0,1,1,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | protocol | Protocol | String | ○ | “tcp” or ”rtu” |
| 2 | node | IP address of PLC device | String | △ | IP address of PLC device, required for “tcp”. |
| 3 | port | TCP port number | Integer | △ | TCP port number of PLC device, required for “tcp”. |
| 4 | device | Device file name | String | △ | Serial port connecting PLC equipment, required for “rtu”. |
| 5 | unit | Modbus Unit ID | Integer | △ | Must be for “rtu”. If omitted at "tcp", 255. |
| 6 | address | Registers address to be write data. | Integer *1 | | If omitted, 0. |
| 7 | function | Modbus function code | Integer *1 | ○ | 5:write_single_coil 15:write_multiple_coils |
| 8 | values | Body of the write data | Integer array | ○ | 0 or 1. If the function key is 5, write the first 1 bit. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (writing to digital input by RTU protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:” 0408f69db38b4d89f25d026d6d9449b7”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “address”:”0x0ce”,
        “function”:15,
        “values”:[ 0,0,0,1,1,1,0,1,1,0,1,1,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Requested value.<br>“tcp” or ”rtu” |
| 5 | node | IP address of PLC device | String | | Requested value.<br>“tcp” only |
| 6 | port | TCP port number | Integer | | Requested value.<br>“tcp” only |
| 7 | device | Device file name | String | | Requested value.<br>“rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | Requested value. |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was wrtten | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | Integer | ○ | Requested value. |
| 12 | values | Body of the written data. | Integer array | ○ | Requested value. |

■Request message sample (reading slave ID)

```
{
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “function”:17
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | protocol | Protocol | String | ○ | ”rtu”only |
| 2 | device | Device file name | String | ○ | |
| 3 | unit | Modbus Unit ID | Integer | ○ | |
| 4 | function | Modbus function code | Integer *1 | ○ | 17:report_slave_id |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Reply message sample (read slave ID)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”72cf056269d6bcd150df8125fbe04710”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “function”:17,
        “values”:[ 7,12]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Requested value. |
| 7 | device | Device file name | String | | Requested value. |
| 8 | unit | Modbus Unit ID | Integer | ○ | Requested value. |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 11 | function | Modbus function code | Integer | ○ | Requested value. |
| 12 | values | List of Modbus Unit IDs that connected. | Integer array | ○ | |

■Response message sample (on error)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:” 7408f69d838b4d89f257036d6d9449b7”,
        “result”:”not queuing”,
        “reason”:“not specified 'function' at least”
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data. | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5) | String | ○ | |
| 3 | result | Status | String | ○ | ”not queuing”: Incorrect request message.<br>“failed”: Faild to connect to PLC equipment, etc. |
| 4 | reason | Reason for error. | String | ○ | |

# 3.2.Modbus Server(Modbus slave)

## 3.2.1.Write operation from PLC device

■Data sample (write to register input by TCP protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “protocol”:”tcp”,
        “node”:”172.16.7.240”,
        “port”:502,
        “unit”:255,
        “memo”:”PLC Server 01”,
        “address”:31,
        “function”:6,
        “values”:[5678]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data. | String | ○ | ISO8601 extended format |
| 2 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or “rtu”.. |
| 3 | node | IP address of listen. | String | | Value set in Web UI.<br>“tcp”only |
| 4 | port | TCP port number | Integer | | ficed 502, tcp”only. |
| 5 | device | Device file name | String | | Value set in Web UI.<br>“rtu”only |
| 6 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 7 | memo | Notes | String | ○ | Value set in Web UI. |
| 8 | address | Registers address from which data was wrtten. | Integer | ○ | 0 to (2048 - registers) |
| 9 | function | Modbus function code | Integer | ○ | 6: write single register<br>16:write multiple registes |
| 10 | values | Body of the written data. | Integer aray | ○ | Unsigned 16bits.<br>The number of arrays varies according to the number of registers written. |

■Data sample (writing to digital input by RTU protocol)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “memo”:”PLC Server 01”,
        “address”:37,
        “function”:5,
        “values”:[1]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data. | String | ○ | ISO8601 extended format |
| 2 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or “rtu”. |
| 3 | node | IP address of listen. | String | | Value set in Web UI.<br>“tcp” only |
| 4 | port | TCP port number | Integer | | fixed 502, “tcp”only. |
| 5 | device | Device file name | String | | Value set in Web UI.<br>“rtu” only |
| 6 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 7 | memo | Notes | String | ○ | Value set in Web UI. |
| 8 | address | Registers address from which data was wrtten. | Integer | ○ | 0 to (2048 - bits) |
| 9 | function | Modbus function code | Integer | ○ | 6: write single register<br>16:write multiple registes |
| 10 | values | Body of the written data. | Integer aray | ○ | 0 or 1.<br>The number of arrays varies according to the number of bits written. |

## 3.2.2.On demand operation from the cloud

■Request message sample (reading register output or register input)

```
{
        “function”:3,
        “address”:31,
        “number”:5,
        “data_type”:”uint16_t”
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | function | Modbus function code | Integer *1 | ○ | 3: read holding registers<br>4: read input registers |
| 2 | address | Registers address to be read data. | Integer *1 | | If omitted, 0. |
| 3 | number | Number of register to be read. | Integer *1 | | If omitted, 1. |
| 4 | data_type | Data type | String | | “uint16_t” :<br>Unsigned 16bits<br>“int16_t”:<br>Signed16bits<br>“uint32lsb_t”:<br>Unsigned 32bits LSB<br>“uint32msb_t”:<br>Unsigned 32bits MSB<br>“int32lsb_t”:<br>Signed 32bits LSB<br>“int32msb_t”:<br>Signed 32bits MSB<br>If omitted,”uint16_t”. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (reading register output or register input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”e4f87480e871555105cc81aac50e5e54”,
        “result”:”done”,
        “protocol”:”tcp”,
        “node”:”172.16.7.249”,
        “port”:502,
        “unit”:255,
        “memo”:”PLC Server 01”,
        “address”:31,
        “function”:3,
        “data_type”:”uint16_t”,
        “values”:[2,0,1234,5678,9876]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Value set in Web UI. “tcp” or ”rtu” |
| 5 | node | IP address of OpenBlocks | String | | “tcp” only |
| 6 | port | TCP port number | Integer | | ficed 502, tcp” only. |
| 7 | device | Device file name | String | | Value set in Web UI. ”rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was read. | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | integer | ○ | Requested value. |
| 12 | data_type | Data type | String | ○ | Requested value. |
| 13 | values | Body of the data. | Integer array | ○ | The number of arrays is variable according to the number of registers requested with the number key. |

* The values of protocol, node, port, device, unit, memo are the values set for the device having the device number of the UNIX domain socket that received the request message.

■Request message sample (reading digital output or digital input)

```
{
        “function”:2
        “address”:37,
        “number”:20
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | function | Modbus function code | Integer *1 | ○ | 1: read coils<br>2: read discrete inputs |
| 2 | address | Registers address to be read data. | Integer *1 | | If omitted, 0. |
| 3 | number | Number of bits to be read. | Integer *1 | | If omitted, 1. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (reading digital output or digital input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”e5910e15403f5e2158a5776cd7136eeb”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “memo”:”PLC04”,
        “address”:37,
        “function”:2,
        “values”:[1,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1,0,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 5 | node | IP address of OpenBlocks | String | | “tcp” only |
| 6 | port | TCP port number | Integer | | ficed 502, tcp” only. |
| 7 | device | Device file name | String | | Value set in Web UI.<br>”rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was read. | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | integer | ○ | Requested value. |
| 12 | values | Body of the data. | Integer array | ○ | 0 or 1.<br>The number of arrays is variable according to the number of bits requested with the number key. |

* The values of protocol, node, port, device, unit, memo are the values set for the device having the device number of the UNIX domain socket that received the request message.

■Request message sample (write to register output or register input)

```
{
        “function”:16,
        “address”:”0x0ab”,
        “function”:16,
        “data_type”:”uint32lsb_t”,
        “values”:[42949672951,21474836471]
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | function | Modbus function code | Integer *1 | ○ | 6:write_single_register<br>10:write_single_input_ registers<br>16:write_multiple_registers<br>20:write_multiple_input_ registers<br>23:write_and_read_registers |
| 2 | address | Registers address to be write data. | Integer *1 | | If omitted, 0. |
| 3 | data_type | Data type | String | | “uint16_t” : Unsigned 16bits<br>“int16_t”: Signed16bits<br>“uint32lsb_t”: Unsigned 32bits LSB<br>“uint32msb_t”: Unsigned 32bits MSB<br>“int32lsb_t”: Signed 32bits LSB<br>“int32msb_t”: Signed 32bits MSB<br>If omitted,”uint16_t”. |
| 4 | values | Body of the write data | Integer array | ○ | If the function key is 6, write the first 1 register. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

*Of the functions, 10:write_single_input_registers and 20:write_multiple_input_registes is a function that does not exist in the original Modubus protocol.

■Response message sample (write to register output or register input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:” 35cf8fa6243d87e0ebb0c2aaaf8eeecf”,
        “result”:”done”
        “protocol”:”tcp”,
        “node”:”172.16.7.249”,
        “port”:502,
        “unit”:255,
        “address”:”0x0ab”,
        “function”:16,
        “data_type”:”uint32lsb_t”,
        “values”:[42949672951,21474836471]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done". |
| 4 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 5 | node | IP address of OpenBlocks | String | | “tcp” only |
| 6 | port | TCP port number | Integer | | ficed 502, tcp”only. |
| 7 | device | Device file name | String | | Value set in Web UI.<br>”rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was wrtten | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | Integer | ○ | Requested value. |
| 12 | data_type | Data type | String | ○ | Requested value. |
| 13 | values | Body of the written data. | Integer array | ○ | Requested value. |

* The values of protocol, node, port, device, unit, memo are the values set for the device having the device number of the UNIX domain socket that received the request message.

■Request message sample (writing to digital output or digital input)

```
{
        “function”:15,
        “address”:”0x0ce”,
        “values”:[0,0,0,1,1,1,0,1,1,0,1,1,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | function | Modbus function code | Integer *1 | ○ | 5:write_single_coil<br>9:write_single_discrete_input<br>15:write_multiple_coils<br>19:write_multiple_discrete_input |
| 2 | address | Registers address to be write data. | Integer *1 | | If omitted, 0. |
| 3 | values | Body of the write data | Integer array | ○ | 0 or 1.<br>If the function key is 5, write the first 1 bit. |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

*Of the functions, 9:write_single_discrete_input and 19 write_multiple_discrete_input is a function that does not exist in the original Modubus protocol.

■Response message sample (writing to digital output or digital input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”c4348e30643dac56cb61bac9743729e7”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “address”:”0x0ce”,
        “function”:15,
        “values”:[ 0,0,0,1,1,1,0,1,1,0,1,1,0,1,1]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 5 | node | IP address of OpenBlocks | String | | “tcp” only |
| 6 | port | TCP port number | Integer | | ficed 502, tcp”only. |
| 7 | device | Device file name | String | | Value set in Web UI.<br>”rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | address | Registers address from which data was wrtten | Integer | ○ | Requested value. |
| 11 | function | Modbus function code | Integer | ○ | Requested value. |
| 12 | values | Body of the written data. | Integer array | ○ | Requested value. |

* The values of protocol, node, port, device, unit, memo are the values set for the device having the device number of the UNIX domain socket that received the request message.

■Request message sample (reading slave ID)

```
{
        “function”:17
}
```

| # | JSON Key | Description | Data Type | Required | Remarks |
|---|---|---|---|---|---|
| 1 | function | Modbus function code | Integer *1 | ○ | 7:report_slave_id |

*1 Hexadecimal notation starting with "0x" in string data type is also possible.

■Response message sample (writing to digital output or digital input)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”e553cae505e64e305373c73d7dd6cd31”,
        “result”:”done”
        “protocol”:”rtu”,
        “device”:”/dev/ttyRS485”,
        “unit”:21,
        “function”:17,
        “values”:[ 21,255]
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5). | String | ○ | |
| 3 | result | Acquisition status | String | ○ | If it succeeds, "done" |
| 4 | protocol | Protocol | String | ○ | Value set in Web UI.<br>“tcp” or ”rtu” |
| 5 | node | IP address of OpenBlocks | String | | “tcp” only |
| 6 | port | TCP port number | Integer | | ficed 502, tcp”only. |
| 7 | device | Device file name | String | | Value set in Web UI.<br>”rtu” only |
| 8 | unit | Modbus Unit ID | Integer | ○ | fixed 255 for "tcp", Value set from Web UI for "rtu". |
| 9 | memo | Notes | String | ○ | Value set in Web UI. |
| 10 | function | Modbus function code | Integer | ○ | Requested value. |
| 11 | values | List of Modbus Unit IDs that connected. | Integer array | ○ | |

* The values of protocol, node, port, device, unit, memo are the values set for the device having the device number of the UNIX domain socket that received the request message.

■Response message sample (on error)

```
{
        “time”:”2017-02-03T14:44:37.020+09:00”,
        “reply_to”:”7408f69d838b4d89f257036d6d9449b7”,
        “result”:”not queuing”,
        “reason”:“not specified 'function' at least”
}
```

| # | JSON Key | Description | Data Type | Always | Remarks |
|---|---|---|---|---|---|
| 1 | time | Timestamp of data. | String | ○ | ISO8601 extended format |
| 2 | reply_to | Hash value of request message (MD5) | String | ○ | |
| 3 | result | Status | String | ○ | ”not queuing”: Incorrect request message.<br>“failed”: Faild to connect to PLC equipment, etc. |
| 4 | reason | Reason for error. | String | ○ | |

Plat'Home Co., Ltd.
NIHON BUILDING KUDANBEKKAN,3F
4-2-3,Kudankita,Chiyoda-ku,TOKYO 102-0073 JAPAN