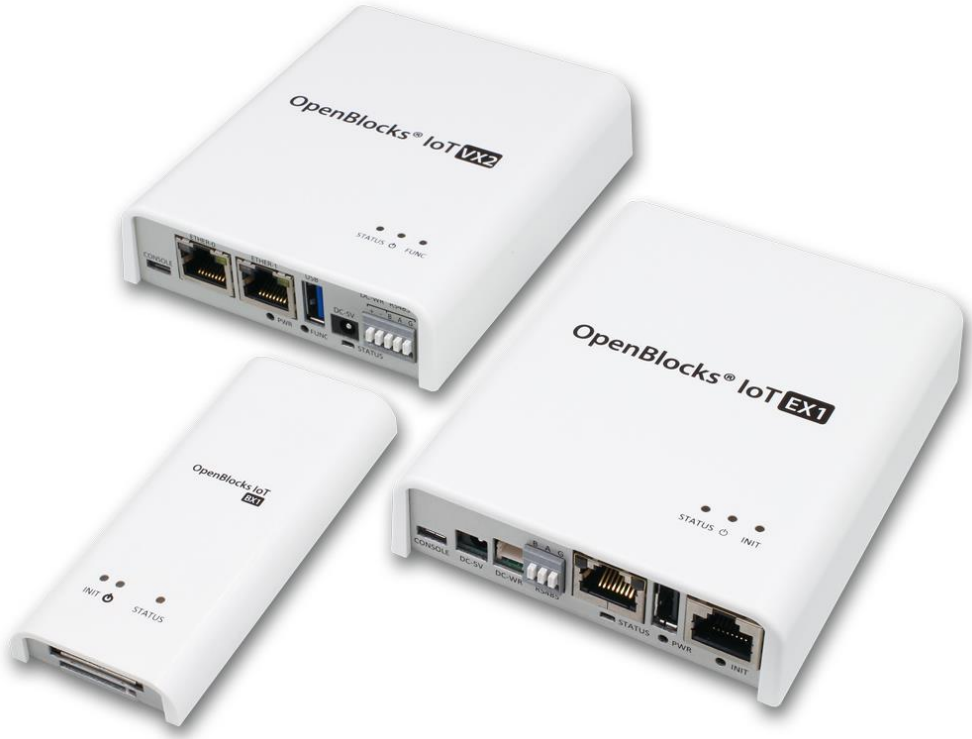
Plat' Home
TECHNOLOGY to serve you.

OpenBlocks IoT Family向け
PD　Handler　JSON
フォーマット一覧



Ver.3.4.0

ぷらっとホーム株式会社

■ 商標について

・ 文中の社名、商品名等は各社の商標または登録商標である場合があります。
・ その他記載されている製品名などの固有名詞は、各社の商標または登録商標です。

■ 使用にあたって

・ 本書の内容の一部または全部を、無断で転載することはご遠慮ください。
・ 本書の内容は予告なしに変更することがあります。
・ 本書の内容については正確を期するように努めていますが、記載の誤りなどにご指摘がございましたら弊社サポート窓口へご連絡ください。
  また、弊社公開の WEB サイトにより本書の最新版をダウンロードすることが可能です。
・ 本装置の使用にあたっては、生命に関わる危険性のある分野での利用を前提とされていないことを予めご了承ください。
・ その他、本装置の運用結果における損害や逸失利益の請求につきましては、上記にかかわらずいかなる責任も負いかねますので予めご了承ください。

目次

1. PD Handler BLE (Node.js)

1.1. Beacon

1.1.1. ビーコン送受信設定

■データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.1.2. BLE デバイス情報送信設定

■データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
3	memo	メモ	String		WEB UI から設定された値

1.2. Sensor

1.2.1. TI Sensor

1.2.1.1. BLE デバイス情報送信設定

■データサンプル


```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	humidity	湿度	Double	△	[%]
4	temperature	温度	Double	△	[℃]
5	accelX	X 方向加速度	Double	△	[G]
6	accelY	Y 方向加速度	Double	△	[G]
7	accelZ	Z 方向加速度	Double	△	[G]
8	gyroX	X 方向角速度	Double	△	[°/s]
9	gyroY	Y 方向角速度	Double	△	[°/s]
10	gyroZ	Z 方向角速度	Double	△	[°/s]
11	magX	X 方向地磁気	Double	△	[μT]
12	magY	Y 方向地磁気	Double	△	[μT]
13	magZ	Z 方向地磁気	Double	△	[μT]
14	pressure	気圧	Double	△	[hPa]
15	objectTemp	物体温度	Double	△	[℃]
16	ambientTemp	周辺温度	Double	△	[℃]
17	lux	照度	Double	△	[lux]
18	memo	メモ	String		WEB UI から設定された値

※センサー依存の JSON キーデータは電池残量や使用モデルにより、含まれない場合があります。

1.2.2. Fujitsu Sensor

1.2.2.1. ビーコン送受信設定

■データサンプル

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	temperature	温度	Double	○	[℃]
10	accelX	X 方向加速度	Double	○	[G]
11	accelY	Y 方向加速度	Double	○	[G]
12	accelZ	Z 方向加速度	Double	○	[G]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.2.2. BLE デバイス情報送信設定

■データサンプル(接続モード、ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	accelX	X 方向加速度	Double	○	[G]
5	accelY	Y 方向加速度	Double	○	[G]
6	accelZ	Z 方向加速度	Double	○	[G]
7	memo	メモ	String		WEB UI から設定された値

1.2.3. ALPS IoT Smart Module

1.2.3.1. ビーコン送受信設定

■データサンプル(ビーコンモード：環境系フォーマット)

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	accelX	X 方向加速度	Double		[G]
10	accelY	Y 方向加速度	Double		[G]
11	accelZ	Z 方向加速度	Double		[G]
12	pressure	気圧	Double		[hPa]
13	humidity	湿度	Double		[%]
14	temperature	温度	Double		[℃]
15	uv	紫外線	Double		[mW/cm2]
16	ambientLight	照度	Double		[lux]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

■データサンプル(ビーコンモード：モーション系フォーマット)

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	accelX	X 方向加速度	Double		[G]
10	accelY	Y 方向加速度	Double		[G]
11	accelZ	Z 方向加速度	Double		[G]
12	geoMagneticX	X 方向地磁気	Double		[uT]
13	geoMagneticY	Y 方向地磁気	Double		[uT]
14	geoMagneticZ	Z 方向地磁気	Double		[uT]
15	pressure	気圧	Double		[hPa]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.3.2. BLE デバイス情報送信設定

■データサンプル(接続モード：データパケット 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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	dataIndex	データインデックス	Integer	○	0~255(シーケンス番号)
4	geoMagneticX	X 方向地磁気	Double		[uT]
5	geoMagneticY	Y 方向地磁気	Double		[uT]
6	geoMagneticZ	Z 方向地磁気	Double		[uT]
7	accelX	X 方向加速度	Double		[G]
8	accelY	Y 方向加速度	Double		[G]
9	accelZ	Z 方向加速度	Double		[G]
10	ms	ミリ秒	Integer	○	
11	second	秒	Integer	○	
12	minute	分	Integer	○	
13	hour	時	Integer	○	
14	memo	メモ	String		WEB UI から設定された値

■データサンプル(接続モード：データパケット 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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	dataIndex	データインデックス	Integer	○	0~255(シーケンス番号)
4	pressure	気圧	Double		[hPa]
5	humidity	湿度	Double		[%]
6	temperature	温度	Double		[℃]
7	uv	紫外線	Double		[mW/cm2]
8	ambientLight	照度	Double		[lux]
9	day	日	Integer	○	
10	month	月	Integer	○	
11	year	年	Integer	○	
12	memo	メモ	String		WEB UI から設定された値

■データサンプル(ビーコンモード：環境系フォーマット)


```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	accelX	X 方向加速度	Double		[G]
4	accelY	Y 方向加速度	Double		[G]
5	accelZ	Z 方向加速度	Double		[G]
6	pressure	気圧	Double		[hPa]
7	humidity	湿度	Double		[%]
8	temperature	温度	Double		[℃]
9	uv	紫外線	Double		[mW/cm2]
10	ambientLight	照度	Double		[lux]
11	memo	メモ	String		WEB UI から設定された値

■データサンプル(ビーコンモード：モーション系フォーマット)


```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	accelX	X 方向加速度	Double		[G]
4	accelY	Y 方向加速度	Double		[G]
5	accelZ	Z 方向加速度	Double		[G]
6	geoMagneticX	X 方向地磁気	Double		[uT]
7	geoMagneticY	Y 方向地磁気	Double		[uT]
8	geoMagneticZ	Z 方向地磁気	Double		[uT]
9	pressure	気圧	Double		[hPa]
10	memo	メモ	String		WEB UI から設定された値

1.2.4. オムロン 環境センサ

1.2.4.1. ビーコン送受信設定

■データサンプル(ビーコンモード：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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	sequence	シーケンス番号	Integer	○	
10	temperature	温度	Double	○	[℃]
11	humidity	湿度	Double	○	[%]
12	light	照度	Integer	○	[lux]
13	uvi	UV インデックス	Double	○	
14	pressure	気圧	Double	○	[hPa]
15	noise	騒音	Double	○	[dB]
16	accelX	X 方向加速度	Double		[G]
17	accelY	Y 方向加速度	Double		[G]
18	accelZ	Z 方向加速度	Double		[G]
19	battery	電池電圧	Integer	○	[mV]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

■データサンプル(ビーコンモード：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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UIから設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	sequence	シーケンス番号	Integer	○	
10	temperature	温度	Double	○	[℃]
11	humidity	湿度	Double	○	[%]
12	light	照度	Integer	○	[lux]
13	uvi	UV インデックス	Double	○	
14	pressure	気圧	Double	○	[hPa]
15	noise	騒音	Double	○	[dB]
16	discomfortIndex	不快指数	Double	○	
17	heatstroke	熱中症危険度	Double	○	[℃]
18	battery	電池電圧	Integer	○	[mV]
ex	ユーザー設定	ユーザー設定内容	String		WEB UIから設定された値

1.2.4.2. BLE デバイス情報送信設定

■データサンプル(接続モード)

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	humidity	湿度	Double	○	[%]
4	temperature	温度	Double	○	[℃]
5	light	照度	Integer	○	[lux]
6	uvi	UV インデックス	Double	○	
7	pressure	気圧	Double	○	[hPa]
8	noise	騒音	Double	○	[dB]
9	discomfortIndex	不快指数	Double	○	
10	heatstroke	熱中症危険度	Double	○	[℃]
11	battery	電池電圧	Integer	○	[mV]
12	memo	メモ	String		WEB UI から設定された値

■データサンプル(ビーコンモード：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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sequence	シーケンス番号	Integer	○	
4	temperature	温度	Double	○	[℃]
5	humidity	湿度	Double	○	[%]
6	light	照度	Integer	○	[lux]
7	uvi	UV インデックス	Double	○	
8	pressure	気圧	Double	○	[hPa]
9	noise	騒音	Double	○	[dB]
10	accelX	X 方向加速度	Double		[G]
11	accelY	Y 方向加速度	Double		[G]
12	accelZ	Z 方向加速度	Double		[G]
13	battery	電池電圧	Integer	○	[mV]
14	memo	メモ	String		WEB UI から設定された値

■データサンプル(ビーコンモード：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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sequence	シーケンス番号	Integer	○	
4	temperature	温度	Double	○	[℃]
5	humidity	湿度	Double	○	[%]
6	light	照度	Integer	○	[lux]
7	uvi	UV インデックス	Double	○	
8	pressure	気圧	Double	○	[hPa]
9	noise	騒音	Double	○	[dB]
10	discomfortIndex	不快指数	Double	○	
11	heatstroke	熱中症危険度	Double	○	[℃]
12	battery	電池電圧	Integer	○	[mV]
13	memo	メモ	String		WEB UI から設定された値

1.2.5. オムロン USB 型環境センサ

1.2.5.1. ビーコン送受信設定

■データサンプル(ビーコンモード：Sensor data)

```
{"
        “time":"2018-11-12T15:29:15.276+09:00",
        "deviceId":"f5961b7e0775",
        "memo":" OMRON USB ",
        "sequence":24,
        "temperature":27.11,
        "humidity":40.75,
        "light":171,
        "pressure":1008.283,
        "noise":56.5,
        "etvoc":26,
        "eco2":573
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	sequence	シーケンス番号	Integer	○	
10	temperature	温度	Double	○	[℃]
11	humidity	湿度	Double	○	[%]
12	light	照度	Integer	○	[lux]
13	pressure	気圧	Double	○	[hPa]
14	noise	騒音	Double	○	[dB]
15	etvoc	等価総揮発性有機化合物	Integer		[ppb]
16	eco2	等価 CO2	Integer		[ppm]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.5.2. BLE デバイス情報送信設定

■データサンプル(ビーコンモード：Sensor Data)

```
{
        “time”:” 2018-11-12T15:35:53.893+09:00”,
        “memo”:”OMRON USB”,
        “deviceId”:” f5961b7e0775”,
        “sequence”:82,
        “temperature”:27.53,
        “humidity”:41.27,
        “light”:173,
        “pressure”:1003.256,
        “noise”:49.26,
        "etvoc":268
        "eco2":583
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sequence	シーケンス番号	Integer	○	
4	temperature	温度	Double	○	[℃]
5	humidity	湿度	Double	○	[%]
6	light	照度	Integer	○	[lux]
7	pressure	気圧	Double	○	[hPa]
8	noise	騒音	Double	○	[dB]
9	etvoc	等価総揮発性有機化合物	Integer		[ppb]
10	eco2	等価 CO2	Integer		[ppm]
11	memo	メモ	String		WEB UI から設定された値

1.2.6. ユニ電子 BLE 温湿度センサー(Logtta)

1.2.6.1. ビーコン送受信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	temperature	温度	Double	○	[℃]
10	humidity	湿度	Double	○	[%]
11	battery	バッテリーレベル	Integer	○	[%]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.6.2. BLE デバイス情報送信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	humidity	湿度	Double	○	[%]
5	battery	バッテリーレベル	Integer	○	[%]
6	memo	メモ	String		WEB UI から設定された値

1.2.7.ユニ電子 BLE　CO2 センサー(Logtta CO2)

1.2.7.1.　　ビーコン送受信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	co2	CO2 濃度	Integer	○	[ppm]
10	battery	バッテリーレベル	Integer	○	[%]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.7.2.　　BLE デバイス情報送信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	co2	CO2 濃度	Integer	○	[ppm]
4	battery	バッテリーレベル	Integer	○	[%]
5	memo	メモ	String		WEB UI から設定された値

1.2.8. ユニ電子 BLE 水温センサー(Logtta Water)

1.2.8.1. ビーコン送受信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	appendixInfo	付随情報	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
8	status	ビーコンステータス	String		ビーコン制御タイプにより表示。(“in”または”out”)
9	temperature	温度	Double	○	[℃]
10	battery	バッテリーレベル	Integer	○	[%]
ex	ユーザー設定	ユーザー設定内容	String		WEB UI から設定された値

1.2.8.2. BLE デバイス情報送信設定

■データサンプル(ビーコンモード)

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

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	battery	バッテリーレベル	Integer	○	[%]
5	memo	メモ	String		WEB UI から設定された値

1.2.9. ラトックシステム Bluetooth ホコリセンサー

1.2.9.1. BLE デバイス情報送信設定

■データサンプル

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sensortime	計測日時	String	○	
4	pm25	PM2.5 濃度	Integer	○	[μg/m3]
5	pm10	PM10 濃度	Integer	○	[μg/m3]
6	pressure	気圧	Integer	○	[hPa]
7	temperature	温度	Integer	○	[℃]
8	humidity	湿度	Integer	○	[%]
9	light	照度	Integer	○	[lx]
10	mode	計測データモード	Integer	○	0:連続計測、1:ワンショット計測
11	memo	メモ	String		WEB UI から設定された値

1.2.10. ラトックシステム Bluetooth エアクオリティモニタ

ー

1.2.10.1. BLE デバイス情報送信設定

■データサンプル

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sensortime	計測日時	String	○	
4	pm25	PM2.5 濃度	Integer	○	[μg/m3]
5	pm10	PM10 濃度	Integer	○	[μg/m3]
6	uvi	UV インデックス	Integer	○	
7	temperature	温度	Double	○	[℃]
8	humidity	湿度	Double	○	[%]
9	phumidity	湿度	Double	○	[%]
10	pressure	気圧	Double	○	[hPa]
11	initstate	初期安定状態	String	○	“wait”または”stability”
12	startstate	起動安定状態	String	○	“wait”または”stability”
13	light	照度	Integer	○	[lx]
14	tvoc	TVOC	Integer	○	[ppb]
15	eco2	eCO2	Integer	○	[ppm]
16	mode	計測データモード	Integer	○	0:連続計測、1:ワンショット計測
17	memo	メモ	String		WEB UI から設定された値

1.2.11. ラトックシステム ワットチェッカー

1.2.11.1. BLE デバイス情報送信設定

■データサンプル

```
{
        “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 キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	sensortime	計測日時	String	○	
4	current	電流値	Double	○	[mA]
5	voltage	電圧	Integer	○	[V]
6	power_consumption	消費電力	Integer	○	[W]
7	memo	メモ	String		WEB UI から設定された値

1.2.12. エレックス工業 μPRISM

■データサンプル


```
{
        “deviceId”:” 00089c161b39”,
        “memo”:” μ Prism”,
        “time”:”2018-09-09T09:55:18.65+09:00”,
        "dataIndex":"111",
        “geoMagneticX”:-10.9,
        “geoMagneticY”:-24.6,
        “geoMagneticZ”:-58.7,
        “accelX”:0.011,
        “accelY”:0.004,
        “accelZ”:1.042,
        “pressure”:1021.84,
        “humidity”:50,38,
        “temperature”:28,5,
        “uvi”:0,1,
        “ambientLight”:404,8,
        “ms”:65,
        “second”:18,
        “minute”:55,
        “hour”:9,
        “day”:9,
        “month”:9,
        “year”:18
}
```


#	JSON キー	内容	データ型	常駐	補足
1	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	dataIndex	データインデックス	Integer	○	
4	geoMagneticX	X 方向地磁気	Double	○	[uT]
5	geoMagneticY	Y 方向地磁気	Double	○	[uT]
6	geoMagneticZ	Z 方向地磁気	Double	○	[uT]
7	accelX	X 方向加速度	Double	○	[G]
8	accelY	Y 方向加速度	Double	○	[G]
9	accelZ	Z 方向加速度	Double	○	[G]
10	pressure	気圧	Double	○	[hPa]
11	humidity	湿度	Double	○	[%RH]
12	temperature	温度	Double	○	[℃]
13	uvi	UV インデックス	Double	○	
14	ambientLight	照度	Double	○	[lx]
15	ms	ミリ秒	Integer	○	
16	second	秒	Integer	○	
17	minute	分	Integer	○	
18	hour	時	Integer	○	
19	day	日	Integer	○	
20	month	月	Integer	○	
21	year	年	Integer	○	
22	memo	メモ	String		WEB UI から設定された値

2. PD Handler BLE (C) with Lua

* ログに出力される JSON データは順不同です。

PD Handler BLE (C) with Lua は、ファームウェア 3.x のハンドラーです。

2.1. Beacon

1 PD Handler BLE with Node.js, 1.1 Beacon を参照ください。

2.2. Sensor

接続モードのセンサには対応していません。

対応センサは 1 PD Handler BLE with Node.js, 1.2 Sensor のビーコンモードを参照ください。

また、BLE デバイス情報送信設定の場合にも rssi 情報が付与されます。

以下、PD Handler BLE with Lua で追加のデバイスのみ記載します。

2.2.1.ナカヨ　呼出しボタン

2.2.1.1.　　ビーコン送受信設定

■データサンプル

```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:” fc97c1aef545”,
        “rssi”:-68
        “uuid”: “a9030100147848 24b2988e6823cfdefa”,
        “major”:”00c8”,
        “minor”:”ffe0”,
        “push”:0
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
3	appendixInfo	付随情報	String		WEB UIから設定された値
4	rssi	受信信号強度	Integer	○	
5	type	ビーコン種別	String		WEB UI にて表示設定
6	data	ペイロードデータ	String		16 進ダンプデータ
7	localname	ローカル名	String		WEB UI にて表示設定
5	uuid	uuid	String	○	
6	major	major	String	○	
7	minor	minor	String	○	
8	push	ボタン押下	Integer	○	0, 1, 2, 3
ex	ユーザー設定	ユーザー設定内容	String		WEB UIから設定された値

2.2.1.2. BLE デバイス情報送信設定

■データサンプル

```
{
        “time”:”2017-12-08T12:34:56.789+09:00”,
        “deviceId”:” fc97c1aef545”,
        “memo”:”Nakayo”,
        “rssi”:-68
        “uuid”: “a9030100147848 24b2988e6823cfdefa”,
        “major”:”00c8”,
        “minor”:”ffe0”,
        “push”:0
}
```

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	deviceId	デバイス ID	String	○	デバイスアドレスから”:”を除き、小文字化した値
3	memo	メモ	String		WEB UI から設定された値
4	rssi	受信信号強度	Integer	○	
5	uuid	uuid	String	○	
6	major	major	String	○	
7	minor	minor	String	○	
8	push	ボタン押下	Integer	○	0, 1, 2, 3

3. PD Handler UART

*ログに出力される JSON データは順不同です。

3.1. EnOcean with Lua

Lua は、ファームウェア 3.x のみ対応です。

3.1.1. 人感センサー(EEP：A50701)

■データサンプル

```
{
        “deviceId”: “0400197A”,
        “time”: “2016-03-14T16:17:02.269+09:00”,
        “svc”: 4.7647,
        “pirs”: “on”,
        “EEP”: “A50701”,
        “memo”: “Occupancy Sensor”,
        “rssi”: -71
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	svc	供給電圧	Double		[V]
4	pirs	検知結果	String	○	“on”または”off”
5	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
6	memo	メモ	String		WEB UI にて設定した値
7	rssi	受信信号強度	Integer	○	

3.1.2. あけしめセンサー(EEP：D50001)

■データサンプル

```
{
        “deviceId”: “04000A1B”,
        “time”: “2016-03-14T16:16:52.525+09:00”,
        “contact”: 0,
        “EEP”: “D50001”,
        “memo”: “Contacts and Switches”,
        “rssi”: -65
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	contact	開閉ステータス	Integer	○	0：Open , 1：Closed
4	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
5	memo	メモ	String		WEB UI にて設定した値
6	rssi	受信信号強度	Integer	○	

3.1.3. 温度センサー(EEP：A50205)

■データサンプル

```
{
        “deviceId”: “04000C66”,
        “time”: “2016-03-14T16:16:59.958+09:00”,
        “temperature”: 25.25,
        “EEP”: “A50205”,
        “memo”: “Temperature Sensors”,
        “rssi”: -82
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
5	memo	メモ	String		WEB UI にて設定した値
6	rssi	受信信号強度	Integer	○	

3.1.4. 温湿度センサー(EEP：A50402)

■データサンプル

```
{
        “deviceId”: “0400267B”,
        “time”: “2017-08-31T14:26:39.283+09:00”,
        “temperature”: 28.16,
        “humidity”: 62.20,
        “EEP”: “A50402”,
        “memo”: “Temperature and Humidity Sensor”,
        “rssi”: -66
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	humidity	湿度	Double	○	[%]
5	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
6	memo	メモ	String		WEB UI にて設定した値
7	rssi	受信信号強度	Integer	○	

3.1.5. 温湿度センサー(EEP：A50403)

■データサンプル

```
{
        “deviceId”: “040005C6”,
        “time”: “2016-03-14T16:15:58.904+09:00”,
        “temperature”: 25.12,
        “humidity”: 35.68,
        “EEP”: “A50403”,
        “memo”: “Temperature and Humidity Sensor”,
        “rssi”: -59
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	temperature	温度	Double	○	[℃]
4	humidity	湿度	Double	○	[%]
5	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
6	memo	メモ	String		WEB UI にて設定した値
7	rssi	受信信号強度	Integer	○	

3.1.6.2 相式 CT センサー(EEP：A51201)

■データサンプル

```
{
        “deviceId”: “0400AE56”,
        “time”: “2016-03-14T16:15:58.904+09:00”,
        “electricity”: 15.0,
        “dataType”:”W”,
        “EEP”: “A51201”,
        “memo”: “Automated Meter Reading (AMR)”,
        “rssi”: -87
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	electricity	電流/電圧/電力値	Double	○	現在値または累積値
4	dataType	データタイプ	String	○	“kWh”または”W”
5	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
6	memo	メモ	String		WEB UI にて設定した値
7	rssi	受信信号強度	Integer	○	

3.1.7.3 相式 CT センサー(EEP：D23202)

■データサンプル

```
{
        “deviceId”: “04015100”,
        “time”: “2016-03-14T16:15:58.904+09:00”,
        “channel1”: 30.0,
        “channel2”: 15.0,
        “channel3”: 10.0,
        “EEP”: “D23202”,
        “memo”: “A.C. Current Clamp”,
        “rssi”: -63
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	channel1	チャネル 1 電流値	Double	○	[A] ※現在値
4	channel2	チャネル 2 電流値	Double	○	[A] ※現在値
5	channel3	チャネル 3 電流値	Double	○	[A] ※現在値
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	

3.1.8. 大気圧センサー(EEP：A50501)

■データサンプル


```
{
　　“deviceId”: “0401520B”,
“time”: “2016-06-07T15:58:22.927+09:00”,
“barometer”: 1010.85,
“telegram_type”: “Hearbeat”,
“EEP”: “A50501”,
“memo”: “barometer”,
“rssi”: -71
}
```


#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	barometer	気圧	Double	○	[hPa]
4	telegram_type	電文タイプ	String	○	“Hearbeat”または”Event triggered”
5	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
6	memo	メモ	String		WEB UI にて設定した値
7	rssi	受信信号強度	Integer	○	

3.1.9. 照度センサー(EEP：A50601)

■データサンプル


```
{
　　“deviceId”: “04007AF5”,
“time”: “2016-06-07T15:58:28.150+09:00”,
“svc”: 2.7800,
“ill1”: 600.0000,
“ill2”: 300.0000,
“EEP”: “A50601”,
“memo”: “ill”,
“rssi”: -65,
"rs":0
}
```


#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	svc	供給電圧	Double	○	[V]
4	ill1	照度 1	Double	○	[lux]
5	ill2	照度 2	Double	○	[lux]
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	
9	rs	測定範囲	Integer	○	

3.1.10. 照度センサー(EEP：A50602)

■データサンプル

```
{
          “deviceId”: “04004715”,
        “time”: “2016-06-07T15:58:28.150+09:00”,
        “svc”: 2.7800,
        “ill1”: 260.0000,
        “ill2”: 260.0000,
        “EEP”: “A50602”,
        “memo”: “ill”,
        “rssi”: -65,
        "rs":0
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	svc	供給電圧	Double	○	[V]
4	ill1	照度 1	Double	○	[lux]
5	ill2	照度 2	Double	○	[lux]
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	
9	rs	測定範囲	Integer	○	

3.1.11. 照度センサー(EEP：A50605)

■データサンプル

```
{
          “deviceId”: “04004F5F”,
        “time”: “2016-06-07T15:58:28.150+09:00”,
        “svc”: 2.7800,
        “ill1”: 360.0000,
        “ill2”: 380.0000,
        “EEP”: “A50605”,
        “memo”: “ill”,
        “rssi”: -65,
        "rs":0
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	svc	供給電圧	Double	○	[V]
4	ill1	照度 1	Double	○	[lux]
5	ill2	照度 2	Double	○	[lux]
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	
9	rs	測定範囲	Integer	○	

3.1.12. CO2 センサー(EEP：A50904)

■データサンプル

```
{
        “deviceId”: “040004FF”,
        “time”: “2016-06-07T15:34:15.126+09:00”,
        “humidity”: 52.000000,
        “temperature”: 28.000000,
        “concentration”: 690,
        “EEP”: “A50904”,
        “memo”: “CO2”,
        “rssi”: -84
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	humidity	湿度	Double	○	[%]
4	temperature	温度	Double	○	[℃]
5	concentration	CO2 濃度	Integer	○	[ppm]
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	

3.1.13. デジタル入力センサー(EEP：A53005)

■データサンプル

```
{
        “deviceId”: “04002D68”,
        “time”: “2016-06-07T15:44:09.621+09:00”,
        “vdd”: 3.1576,
        “signal_type”: “Heart beat signal”,
        “count”: 127,
        “EEP”: “A53005”,
        “memo”: “button”,
        “rssi”: -58
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	vdd	供給電圧	Double	○	[V]
4	signal_type	信号種別	String	○	“Normal signal”または”Heart beat signal”
5	count	序数	Integer	○	0～127
6	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
7	memo	メモ	String		WEB UI にて設定した値
8	rssi	受信信号強度	Integer	○	

3.1.14. 2 ロッカースイッチ(EEP：F60204)

■データサンプル

```
{
        “deviceId”: “002BC9C8”,
        “time”: “2016-07-26T10:45:09.625+09:00”,
        “ebo”: “pressed”,
        “rbi”: “released”,
        “rbo”: “released”,
        “rai”: “pressed”,
        “rao”: “released”,
        “EEP”: “F60204”,
        “memo”: “2 rocker switch”,
        “rssi”: -70
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	ebo	エネルギー状態	String	○	“pressed”または”released”
4	rbi	ロッカースイッチ B の状態 I	String	○	“pressed”または”released”
5	rbo	ロッカースイッチ B の状態 O	String	○	“pressed”または”released”
6	rai	ロッカースイッチ A の状態 I	String	○	“pressed”または”released”
7	rao	ロッカースイッチ A の状態 O	String	○	“pressed”または”released”
8	EEP	EnOcean プロファイル	String	○	WEB UI にて設定した値
9	memo	メモ	String		WEB UI にて設定した値
10	rssi	受信信号強度	Integer	○	

3.1.15. RAW データ時

■データサンプル

```
{
        “deviceId”: “0400197A”,
        “time”: “2016-03-14T16:45:32.643+09:00”,
        “data”: “55000c020ae66200000400197a1c080b8720013da6”,
        “EEP”: “A50701”,
        “memo”: “raw data”,
        “rssi”: -63
}
```

#	JSON キー	内容	データ型	常駐	補足
1	deviceId	ID	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	data	ペイロード	String	○	
4	EEP	EnOcean プロファイル	String		WEB UI にて設定した値
5	memo	メモ	String		WEB UI にて設定した値
6	rssi	受信信号強度	Integer	○	

3.2. Wi-SUN

3.2.1. 瞬時電力(B ルート)

■データサンプル

```
{
        “address”: “0011223344556677”,
        “date”: “2016-1-25”,
        “time”: “12:34”,
        “inst”: 2147483645
}
```

#	JSON キー	内容	データ型	常駐	補足
1	address	アドレス	String	○	スマートメーター側のアドレス
2	date	対象データ年月日	String	○	
3	time	対象データ時間	String	○	
4	inst	瞬時電力	Integer	○	[W]

3.2.2. 積算電力(B ルート)

■データサンプル

```
{
        “address”: “0011223344556677”,
        “time”: “2016-1-25T12:34:56”,
        “ratio”: 10,
        “unit”: 0.01,
        “cumu”: 2147483645,
        “cumu_re”: -1
}
```

#	JSON キー	内容	データ型	常駐	補足
1	address	アドレス	String	○	スマートメーター側のアドレス
2	time	対象データ日時	String	○	
3	ratio	倍率	Integer	○	0 ～ 999,999
4	unit	単位	Double	○	以下の値。 1 , 0.1 , 0.01 , 0.001 , 0.0001 10 , 100 , 1000 , 10000
5	cumu	正方向累積値	Integer	○	0 ~ 99,999,999。但し、-1 はデータ無し扱い。
6	cumu_re	逆方向累積値	Integer	○	0 ~ 99,999,999。但し、-1 はデータ無し扱い。

※計算方式

- 正方向 積算電力

  <*Total*> = <cumu> ×<ratio> × <unit>

- 逆方向 積算電力

  <*Total*> = <cumu_re> ×<ratio> × <unit>

4. PD Handler HVSMC

* ログに出力される JSON データは順不同です。

4.1. RAW データ

■データサンプル

```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “memo”: “raw data”,
        “datatype”: “raw_data”,
        “raw_data”:[
          {
            “EPC”: “80”,
            “PDC”: 1,
            “EDT”: “30”
          },
          {
            “EPC”: “88”,
            “PDC”: 1,
            “EDT”: “41”
          }
        ]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	memo	メモ	String		WEB UI にて設定した値
4	datatype	データータイプ	String	○	raw_data
5	raw_data		JSON オブジェクト配列	○	
6	EPC	ECHONET プロパティ	String	○	
7	PDC	プロパティデータカウンタ	Integer	○	
8	EDT	ECHONET プロパティ値データ	String	○	

4.2. 動作状態

■データサンプル

```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “80”: “ON”,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “op_stat”
}
```

#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	80	動作状態	String	○	“ON” or “OFF”
4	memo	メモ	String		WEB UI にて設定した値
5	datatype	データータイプ	String	○	op_stat

4.3. 異常発生状態

■データサンプル

```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “88”: “not”,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “fault_stat”
}
```

#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	88	異常発生状態	String	○	“occurred” or “not”
4	memo	メモ	String		WEB UI にて設定した値
5	datatype	fault_stat	String	○	fault_stat

4.4. ECHONET Lite 属性情報

■データサンプル


```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “82”: “F”,
        “9D”: “808188”,
        “9E”: “81E1”,
        “9F”: “808182888A8D97989D9E9FD3D4E0E1E2E3E4E5E6E7C1C2C3C4C5C6C7CACBCCCDCE”,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “echonet_attr”
}
```


#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	82	規格 Version 情報	String	○	
4	9D	状態アナウンスプロパティマップ	String	○	
5	9E	Set プロパティマップ	String	○	
6	9F	Get プロパティマップ	String	○	
7	memo	メモ	String		WEB UI にて設定した値
8	datatype	データータイプ	String	○	echonet_attr

4.5.高圧スマート電力量メータ属性情報

■データサンプル


```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “D3”: 1,
        “D4”: 0.1,
        “E0”: 1,
        “E5”: 8,
        “E6”: 0.001,
        “C4”: 8,
        “C5”: 0.001,
        “C7”: 0.001,
        “CC”: 6,
        “CD”: 0.01,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “hvsm_attr”
}
```


#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	D3	係数	Integer	○	
4	D4	係数の倍率	Double	○	
5	E0	確定日	Integer	○	
6	E5	積算有効電力量有効桁数	Integer	○	
7	E6	積算有効電量量単位	Double	○	[kWh]
8	C4	需要電力有効桁数	Integer	○	
9	C5	需要電力単位	Double	○	[kWh]
10	C7	累積最大需要電力単位	Double	○	[kWh]
11	CC	力測積算無効電力量(遅れ)有効桁数	Integer	○	
12	CD	力測積算無効電力量(遅れ)単位	Double	○	[kvarh]
13	memo	メモ	String		WEB UI にて設定した値
14	datatype	データータイプ	String	○	hvsm_attr

4.6. 定時計測値

■データサンプル


```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-11T12:30:00+09:00”,
        “E3”: 1234567.8,
        “C3”: 1234567.8,
        “CB”: 1234567.8,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “fixed_measured”
}
```


#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	計測年月日時刻	String	○	ISO8601 拡張書式
3	E3	定時積算有効電力量計測値	Double	○	
4	C3	定時需要電力	Double	○	
5	CB	定時力測積算無効電力量(遅れ)計測値	Double	○	
6	memo	メモ	String		WEB UI にて設定した値
7	datatype	データータイプ	String	○	fixed_measured

4.7. 計測値

■データサンプル


```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-11T12:34:56+09:00”,
        “E2”: 1234567.8,
        “E4”: 1234567.8,
        “CA”: 1234567.8,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “measured”
}
```


#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	計測年月日時刻	String	○	ISO8601 拡張書式
3	E2	積算有効電力量計測値	Double	○	
4	E4	力測積算有効電力量計測値	Double	○	
5	CA	力測積算無効電力量(遅れ)計測値	Double	○	
6	memo	メモ	String		WEB UI にて設定した値
7	datatype	データータイプ	String	○	measured

4.8.需要電力

■データサンプル


```
{
        “8D”: “0123456789AB”,
        “time”: “2018-11-21T12:34:56+09:00”,
        “C1”: 1234567.8,
        “C2”: 1234567.8,
        “memo”: “pd-handler-hvsmc”,
        “datatype”: “demand”
}
```


#	JSON キー	内容	データ型	常駐	補足
1	8D	製造番号	String	○	
2	time	データ取得日時	String	○	ISO8601 拡張書式
3	C1	月間最大需要電力	Double	○	
4	C2	累積最大需要電力	Double	○	
5	memo	メモ	String		WEB UI にて設定した値
6	datatype	データータイプ	String	○	demand

5. PD Handler Modbus

5.1. Modbus クライアント(Modbus マスター)

5.1.1. PLC 機器へのポーリング動作時

■データサンプル(TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UI から設定された値 “tcp”又は”rtu”
3	node	取得先 IP アドレス	String	○	WEB UI から設定された値 “tcp”時のみ
4	port	ポート番号	Integer	○	WEB UI から設定された値 “tcp”時のみ
5	device	デバイスファイル名	String		WEB UI から設定された値 “rtu”時のみ
6	unit	Modbus Unit ID	Integer	○	WEB UI から設定された値
7	memo	メモ	String	○	WEB UI から設定された値
8	address	読み込みアドレス	Integer	○	WEB UI から設定された値
9	function	Modbus function code	integer	○	WEB UI から設定された値 3: レジスタ出力 4: レジスタ入力
10	data_type	データの型	String	○	WEB UI から設定された値 “uint16_t”： 符号なし 16bits “int16_t”: 符号付 16bits “uint32lsb_t”: 符号なし 32bits LSB “uint32msb_t”: 符号なし 32bits MSB “int32lsb_t”: 符号付 32bits LSB “int32msb_t”: 符号付 32bits MSB
11	values	読み込み値	Integer 配列	○	配列数は読み込みレジスタ数の設定に応じて可変。

■データサンプル(RTU プロトコルによるデジタル出力又はデジタル入力の読み込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UIから設定された値 “tcp”又は”rtu”
3	node	取得先 IP アドレス	String		WEB UIから設定された値 “tcp”時のみ
4	port	ポート番号	Integer		WEB UIから設定された値 “tcp”時のみ
5	device	デバイスファイル名	String		WEB UIから設定された値 “rtu”時のみ
6	unit	Modbus Unit ID	Integer	○	WEB UIから設定された値
7	memo	メモ	String	○	WEB UIから設定された値
8	address	読み込みアドレス	Integer	○	WEB UIから設定された値
9	function	Modbus function code	Integer	○	WEB UIから設定された値 1: デジタル出力 2: デジタル入力
10	values	読み込み値	Integer 配列	○	0 又は 1。 配列数は読み込みレジスタ数の設定に応じて可変。

5.1.2. クラウドからのオンデマンド動作時

■リクエストメッセージサンプル(TCP プロトコルによるレジスタ出力又はレジスタ入力の読み出し)

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	“tcp”又は”rtu”
2	node	取得先 IP アドレス	String	△	PLC 機器の IP アドレス “tcp”時は必須
3	port	ポート番号	Integer	△	PLC 機器のポート番 “tcp”時は必須
4	device	デバイスファイル名	String	△	PLC 機器を接続するシリアルポート “rtu”時は必須
5	unit	Modbus Unit ID	Integer	△	“rtu”時は必須 “tcp”時の省略時は 255
6	address	読み込みアドレス	Integer *1		読込開始アドレス 省略時は 0
7	function	Modbus function code	Integer *1	○	3: read holding registers 4: read input registers
8	number	読み込むレジスタ数	Integer *1		省略時は 1
9	data_type	データの型	String		“uint16_t”： 符号なし 16bits “int16_t”: 符号付 16bits “uint32lsb_t”: 符号なし 32bits LSB “uint32msb_t”: 符号なし 32bits MSB “int32lsb_t”: 符号付 32bits LSB “int32msb_t”: 符号付 32bits MSB 省略時は”uint16_t”

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(TCP プロトコルによるレジスタ出力又はレジスタ入力の読み込み)


```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	リクエストされた値 “tcp”又は”rtu”
5	node	取得先 IP アドレス	String		リクエストされた値 “tcp”時のみ
6	port	ポート番号	Integer		リクエストされた値 “tcp”時のみ
7	device	デバイスファイル名	String		リクエストされた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String	○	WEB UI から設定さた値
10	address	読み込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	integer	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	読み込み値	Integer 配列	○	配列数は number キーでリクエストされたレジスタ数に応じて可変。

■リクエストメッセージサンプル(RTU プロトコルによるデジタル出力又はデジタル入力の読み込み)

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	“tcp”又は”rtu”
2	node	取得先 IP アドレス	String	△	PLC 機器の IP アドレス “tcp”時は必須
3	port	ポート番号	Integer	△	PLC 機器のポート番 “tcp”時は必須
4	device	デバイスファイル名	String	△	PLC 機器を接続するシリアルポート “rtu”時は必須
5	unit	Modbus Unit ID	Integer	△	“rtu”時は必須 “tcp”時の省略時は 255
6	address	読み込みアドレス	Integer *1		読込開始アドレス 省略時は 0
7	function	Modbus function code	Integer *1	○	1: read coils 2: read discrete inputs
8	number	読み込むビット数	Integer *1		省略時は 1

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(RTU プロトコルによるデジタル出力又はデジタル入力の読み込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	リクエストされた値 “tcp”又は”rtu”
5	node	取得先 IP アドレス	String		リクエストされた値 “tcp”時のみ
6	port	ポート番号	Integer		リクエストされた値 “tcp”時のみ
7	device	デバイスファイル名	String		リクエストされた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String	○	WEB UI から設定された値
10	address	読み込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	Integer	○	リクエストされた値
12	values	読み込み値	Integer 配列	○	0 又は 1。 配列数は number キーでリクエストされたビット数に応じて可変。

■リクエストメッセージサンプル(TCP プロトコルによるレジスタ入力への書き込み)

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	“tcp”又は”rtu”
2	node	取得先 IP アドレス	String	△	PLC 機器の IP アドレス “tcp”時は必須
3	port	ポート番号	Integer	△	PLC 機器のポート番 “tcp”時は必須
4	device	デバイスファイル名	String	△	PLC 機器を接続するシリアルポート “rtu”時は必須
5	unit	Modbus Unit ID	Integer	△	“rtu”時は必須 “tcp”時の省略時は 255
5	address	書き込みアドレス	Integer *1		読込開始アドレス 省略時は 0
6	function	Modbus function code	Integer *1	○	6:write_single_register 16:write_multiple_registers 23:write_and_read_registers
7	data_type	データの型	String		“uint16_t”： 符号なし 16bits “int16_t”: 符号付 16bits “uint32lsb_t”: 符号なし 32bits LSB “uint32msb_t”: 符号なし 32bits MSB “int32lsb_t”: 符号付 32bits LSB “int32msb_t”: 符号付 32bits MSB 省略時は”uint16_t”
8	values	書き込む値	Integer 配列	○	function キーが 6 の場合、は先頭の 1 レジスタを書き込む

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(TCP プロトコルによるレジスタ入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	リクエストされた値 “tcp”又は”rtu”
5	node	取得先 IP アドレス	String		リクエストされた値 “tcp”時のみ
6	port	ポート番号	Integer		リクエストされた値 “tcp”時のみ
7	device	デバイスファイル名	String		リクエストされた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String	○	WEB UI から設定された値
10	address	書き込みアドレス	Integer *1	○	リクエストされた値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	書き込んだ値	Integer 配列	○	リクエストされた値

■リクエストメッセージサンプル(RTU プロトコルによるデジタル入力への書き込み)

```
{
        “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 キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	“tcp”又は”rtu”
2	node	取得先 IP アドレス	String	△	PLC 機器の IP アドレス “tcp”時は必須
3	port	ポート番号	Integer	△	PLC 機器のポート番 “tcp”時は必須
4	device	デバイスファイル名	String	△	PLC 機器を接続するシリアルポート “rtu”時は必須
5	unit	Modbus Unit ID	Integer	△	“rtu”時は必須 “tcp”時の省略時は 255
6	address	書き込みアドレス	Integer *1		読込開始アドレス 省略時は 0
7	function	Modbus function code	Integer *1	○	5:write_single_coil 15:write_multiple_coils
8	values	書き込む値	Integer 配列	○	0 又は 1 function キーが 5 の場合、は先頭の 1 ビットを書き込む

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(RTU プロトコルによるデジタル入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	リクエストされた値 “tcp”又は”rtu”
5	node	取得先 IP アドレス	String		リクエストされた値 “tcp”時のみ
6	port	ポート番号	Integer		リクエストされた値 “tcp”時のみ
7	device	デバイスファイル名	String		リクエストされた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String	○	WEB UI から設定された値
10	address	書き込みアドレス	Integer *1	○	リクエストされた値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	values	書き込んだ値	Integer 配列	○	リクエストされた値

■リクエストメッセージサンプル(スレーブ ID の読み出し)

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

#	JSON キー	内容	データ型	必須	補足
1	protocol	プロトコル	String	○	”rtu”のみ
2	device	デバイスファイル名	String	○	
3	unit	Modbus Unit ID	Integer	○	
4	function	Modbus function code	Integer *1	○	17:report_slave_id

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(スレーブ ID の読み出し)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	リクエストされた値
7	device	デバイスファイル名	String	○	リクエストされた値
8	unit	Modbus Unit ID	Integer	○	リクエストされた値
9	memo	メモ	String	○	WEB UI から設定された値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	values	接続されている Modbus Unit ID の一覧	Integer 配列	○	

■応答メッセージサンプル(エラー時)

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	”not queuing”:リクエストメッセージの不備 “failed”:PLC 機器に接続できなかった場合等
4	reason	エラーの理由	String	○	

5.2. Modbus サーバー(Modbus スレーブ)

5.2.1. PLC 機器からの書き込み動作時

■データサンプル(TCP プロトコルによるレジスタ入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UI から設定された値 “tcp”又は”rtu”
3	node	書き込み元 IP アドレス	String		WEB UI から設定された値 “tcp”時のみ
4	port	ポート番号	Integer		502（固定) “tcp”時のみ
5	device	デバイスファイル名	String		WEB UI から設定された値 “rtu”時のみ
6	unit	Modbus Unit ID	Integer	○	“tcp”時は 255 に固定 “rtu”時は WEB UI から設定された値
7	memo	メモ	String	○	WEB UI から設定された値
8	address	書き込みアドレス	Integer	○	0 ～ (2048 - registers)の範囲
9	function	Modbus function code	Integer	○	6: write single register 16:write multiple registes
10	values	書き込まれた値	Integer 配列	○	16 ビット符号無し整数値。配列数は書き込まれたレジスタ数に応じて可変。

■データサンプル(RTU プロトコルによるデジタル入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	protocol	プロトコル	String	○	WEB UI から設定された値 “tcp”又は”rtu”
3	node	書き込み元 IP アドレス	String		WEB UI から設定された値 “tcp”時のみ
4	port	ポート番号	Integer		502（固定） “tcp”時のみ
5	device	デバイスファイル名	String		WEB UI から設定された値 “rtu”時のみ
6	unit	Modbus Unit ID	Integer	○	“tcp”時は 255 に固定 “rtu”時は WEB UI から設定された値
7	memo	メモ	String	○	WEB UI から設定された値
8	address	書き込みアドレス	Integer	○	0 ～ (2048 - bits)の範囲
9	function	Modbus function code	Integer	○	5: write single 15: write multiple coils
10	values	書き込まれた値	Integer 配列	○	0 又は 1。 配列数は書き込まれたビット数に応じて可変。

5.2.2. クラウドからのオンデマンド動作時

■リクエストメッセージサンプル(レジスタ出力又はレジスタ入力の読み込み)

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

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	Integer *1	○	3: read holding registers 4: read input registers
2	address	読み込みアドレス	Integer *1		読込開始アドレス 省略時は 0
3	number	読み込むレジスタ数	Integer *1		省略時は 1
4	data_type	データの型	String		“uint16_t”： 符号なし 16bits “int16_t”: 符号付 16bits “uint32lsb_t”: 符号なし 32bits LSB “uint32msb_t”: 符号なし 32bits MSB “int32lsb_t”: 符号付 32bits LSB “int32msb_t”: 符号付 32bits MSB 省略時は”uint16_t”

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(レジスタ出力又はレジスタ入力の読み込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	WEB UI から設定さた値 “tcp”又は”rtu”
5	node	サーバ自身の IP アドレス	String		“tcp”時のみ
6	port	ポート番号	Integer		502(固定) “tcp”時のみ
7	device	デバイスファイル名	String		WEB UI から設定さた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値 “tcp”時は 255 に固定
9	memo	メモ	String	○	WEB UI から設定された値
10	address	読み込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	integer	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	読み込み値	Integer 配列	○	配列数は number キーでリクエストされたレジスタ数に応じて可変。

※protocol,node,port,device,unit,memo の値は、リクエストメッセージを受けた UNIX ドメインソケットのデバイス番号を持つデバイスに設定されている値です。

■リクエストメッセージサンプル(デジタル出力又はデジタル入力の読み込み)

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

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	Integer *1	○	1: read coils 2: read discrete inputs
2	address	読み込みアドレス	Integer *1		読込開始アドレス 省略時は 0
3	number	読み込むビット数	Integer *1		省略時は 1

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(デジタル出力又はデジタル入力の読み込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	WEB UI から設定さた値 “tcp”又は”rtu”
5	node	サーバ自身の IP アドレス	String		“tcp”時のみ
6	port	ポート番号	Integer		502(固定) “tcp”時のみ
7	device	デバイスファイル名	String		WEB UI から設定さた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値 “tcp”時は 255 に固定
9	memo	メモ	String	○	WEB UI から設定された値
10	address	読み込みアドレス	Integer	○	リクエストされた値
11	function	Modbus function code	Integer	○	リクエストされた値
12	values	読み込み値	Integer 配列	○	0 又は 1。 配列数は number キーでリクエストされたビット数に応じて可変。

※protocol,node,port,device,unit,memo の値は、リクエストメッセージを受けた UNIX ドメインソケットのデバイス番号を持つデバイスに設定されている値です。

■リクエストメッセージサンプル(レジスタ出力又はレジスタ入力への書き込み)

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

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	Integer *1	○	6:write_single_register 10:write_single_input_ registers 16:write_multiple_registers 20:write_multiple_input_ registers 23:write_and_read_registers
2	address	書き込みアドレス	Integer *1		読込開始アドレス 省略時は 0
3	data_type	データの型	String		“uint16_t”： 符号なし 16bits “int16_t”: 符号付 16bits “uint32lsb_t”: 符号なし 32bits LSB “uint32msb_t”: 符号なし 32bits MSB “int32lsb_t”: 符号付 32bits LSB “int32msb_t”: 符号付 32bits MSB 省略時は”uint16_t”
4	values	書き込む値	Integer 配列	○	function キーが 6 の場合、は先頭の 1 レジスタを書き込む

*1 String 型で”0x”から始まる 16 進数表記も可能

※function の内、10: write_single_input_registers と 20:write_multiple_input_registes は、本来の Modubus プロトコルには存在しない機能です。

■応答メッセージサンプル(レジスタ出力又はレジスタ入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	WEB UI から設定さた値 “tcp”又は”rtu”
5	node	サーバ自身の IP アドレス	String		“tcp”時のみ
6	port	ポート番号	Integer		502(固定) “tcp”時のみ
7	device	デバイスファイル名	String		WEB UI から設定さた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値 “tcp”時は 255 に固定
9	memo	メモ	String	○	WEB UI から設定された値
10	address	書き込みアドレス	Integer *1	○	リクエストされた値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	data_type	データの型	String	○	リクエストされた値
13	values	書き込んだ値	Integer 配列	○	リクエストされた値

※protocol,node,port,device,unit,memo の値は、リクエストメッセージを受けた UNIX ドメインソケットのデバイス番号を持つデバイスに設定されている値です。

■リクエストメッセージサンプル(デジタル出力又はデジタル入力への書き込み)

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

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	Integer *1	○	5:write_single_coil 9:write_single_discrete_input 15:write_multiple_coils 19:write_multiple_discrete_input
6	address	書き込みアドレス	Integer *1		読込開始アドレス 省略時は 0
8	values	書き込む値	Integer 配列	○	0 又は 1 function キーが 5 の場合、は先頭の 1 ビットを書き込む

*1 String 型で”0x”から始まる 16 進数表記も可能

※function の内、9: write_single_discrete_input と 19: write_multiple_discrete_input は、本来の Modubus プロトコルには存在しない機能です。

■応答メッセージサンプル(デジタル出力又はデジタル入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	WEB UI から設定さた値 “tcp”又は”rtu”
5	node	サーバ自身の IP アドレス	String		“tcp”時のみ
6	port	ポート番号	Integer		502(固定) “tcp”時のみ
7	device	デバイスファイル名	String		WEB UI から設定さた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値 “tcp”時は 255 に固定
9	memo	メモ	String	○	WEB UI から設定された値
10	address	書き込みアドレス	Integer *1	○	リクエストされた値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	values	書き込んだ値	Integer 配列	○	リクエストされた値

※protocol,node,port,device,unit,memo の値は、リクエストメッセージを受けた UNIX ドメインソケットのデバイス番号を持つデバイスに設定されている値です。

■リクエストメッセージサンプル(スレーブ ID の読み出し)

```
{
        “function”:17
}
```

#	JSON キー	内容	データ型	必須	補足
1	function	Modbus function code	Integer *1	○	7:report_slave_id

*1 String 型で”0x”から始まる 16 進数表記も可能

■応答メッセージサンプル(デジタル出力又はデジタル入力への書き込み)

```
{
        “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 キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	成功時は”done”
4	protocol	プロトコル	String	○	WEB UI から設定さた値 “tcp”又は”rtu”
5	node	サーバ自身の IP アドレス	String		“tcp”時のみ
6	port	ポート番号	Integer		502(固定) “tcp”時のみ
7	device	デバイスファイル名	String		WEB UI から設定さた値 “rtu”時のみ
8	unit	Modbus Unit ID	Integer	○	WEB UI から設定さた値 “tcp”時は 255 に固定
9	memo	メモ	String	○	WEB UI から設定された値
11	function	Modbus function code	Integer *1	○	リクエストされた値
12	values	サーバ自身に設定さている Modbus Unit ID の一覧	Integer 配列	○	

※protocol,node,port,device,unit,memo の値は、リクエストメッセージを受けた UNIX ドメインソケットのデバイス番号を持つデバイスに設定されている値です。

■応答メッセージサンプル(エラー時)

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

#	JSON キー	内容	データ型	常駐	補足
1	time	データ取得日時	String	○	ISO8601 拡張書式
2	reply_to	リクエストメッセージの MD5 値	String	○	
3	result	取得ステータス	String	○	”not queuing”:リクエストメッセージの不備 “failed”:PLC 機器に接続できなかった場合等
4	reason	エラーの理由	String	○	

6. PD Handler Modbus 2

6.1. Modbus クライアント(Modbus マスター)

6.1.1. PLC 機器へのポーリング動作時

■データサンプル

```
{
    "timestamp": "2019-11-15T10:02:17.277+09:00",
    "unitId": 1,
    "maker": "OMRON",
    "product": "Smart Power Monitor",
    "model": "KM-N1-FLK",
    "sku": "00",
    "voltage1": 103.6,
    "voltage2": 0.0,
          :
}
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

6.1.2. クラウドからのオンデマンド動作時

■リクエストメッセージサンプル(ポーリング中のレジスタに対するオンデマンド読み込み)

```
{
        “unitId”:1
}
```

#	JSON キー	内容	データ型	必須	補足
1	unitId	Modbus Unit ID	Integer or String	○	

■応答メッセージサンプル

```
{
        "timestamp" : "2019-11-15T10:19:54.651+09:00",
        "unitId" : 1,
        "maker" : "OMRON",
        "product" : "Smart Power Monitor",
        "model" : "KM-N1-FLK",
        "sku" : "00",
        "request_from" : "0x01(0)",
        "reply_to" : "661d94902a592172865e6853159dc138",
        “result”: true,
        "voltage1" : 103.5,
        "voltage2" : 0,
          :
}
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		
7	request_from	リクエスト元クラウド ID	String		reply が true に設定されている時のみ表示
8	reply_to	リクエストメッセージの MD5 値	String		
9	result	取得ステータス	Boolena		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

■応答メッセージサンプル(エラー時)

```
{
   "timestamp" : "2019-11-15T11:34:54.679+09:00",
   "request_from" : "0x01(0)",
   "reply_to" : "8473b3a74dfe726d256314fcfd1186cf",
   "result" : false,
   “reason" : "parser_sub_socket(): unitId 2 not found"
   }
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	request_from	リクエスト元クラウド ID	String	○	
3	reply_to	リクエストメッセージの MD5 値	String	○	
4	result	取得ステータス	Boolena	○	失敗時は false
5	reason	エラーの理由	String	○	

reply が true に設定されている時のみ、応答を返します。

■リクエストメッセージサンプル(レジスタに対するオンデマンド書き込み)

```
{
    "unitId" : 1,
    "rawRequest" : {
        "configMode" : null
    },
    "write" : {
        "ctType" : 1
    }
}
```

#	JSON キー	内容	データ型	必須	補足
1	unitId	Modbus Unit ID	Integer or String	○	
2	rawRequest	RAW リクエスト	JSON obj		モデリングファイルの設定に応じて異なります。
3	write	書込み制御	JSON obj		モデリングファイルの設定に応じて異なります。

■応答メッセージサンプル

```
{
    "timestamp" : "2019-11-18T10:07:46.587+09:00",
    "unitId" : 1,
    "maker" : "OMRON",
    "product" : "Smart Power Monitor",
    "model" : "KM-N1-FLK",
    "sku" : "00",
    "request_from" : "0x01(0)",
    "reply_to" : "dd8d548a7adf42102a6a6762d0866520",
    "result" : true,
    "configMode" : "0x0106FFFF07008BDE",
    "ctType" : 1
}
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		
7	request_from	リクエスト元クラウド ID	String		reply が true に設定されている時のみ表示
8	reply_to	リクエストメッセージの MD5 値	String		
9	result	取得ステータス	Boolean		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

rawRequest と write を同時にリクエストした場合で、いずれか片方のみがエラーとなった場合、エラーとなったリクエストに対する応答のみが返されます。

6.2. Modbus サーバー(Modbus スレーブ)

6.2.1. PLC 機器からの書き込み動作時

■データサンプル(TCP プロトコルによるレジスタ出力への書き込み)

```
{
    "timestamp" : "2019-11-22T10:34:49.179+09:00",
    "unitId" : 255,
    "maker" : "PlatHome Co., Ltd.",
    "product" : "Virtual PLC server",
    "model" : "32 bits 8 port Register",
    "sku" : "0000"
    "outRegs" : [
        "0x00000001",
        "0x00000002",
        "0x00000003",
        "0x00000004",
        "0x00000005",
        "0x00000006",
        "0x00000007",
        "0x00000008"
    ]
}
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

6.2.2. クラウドからのオンデマンド動作時

■リクエストメッセージサンプル(レジスタの読み込み)

```
{}
```

レジスタの読み込みのためのリクエストメッセージは、空の JSON オブジェクトです。

■応答メッセージサンプル

```
{
    "timestamp" : "2019-11-22T11:28:50.084+09:00",
    "unitId" : 255,
    "maker" : "PlatHome Co., Ltd.",
    "product" : "Virtual PLC server",
    "model" : "32 bits 8 port Register",
    "sku" : "0000",
    "request_from" : "0x01(0)",
    "reply_to" : "99914b932bd37a50b983c5e7c90ae93b",
    "result" : true,
    "outBits" : "0x0",
    "inBit0" : 0,
    "inBit1" : 0,
    "inBit2" : 0,
    "inBit3" : 0,
    "outRegs" : [
        "0x00000001",
        "0x00000002",
        "0x00000003",
        "0x00000004",
        "0x00000005",
        "0x00000006",
        "0x00000007",
        "0x00000008"
    ],
    "inReg0" : 0,
    "inReg1" : 0,
    "inReg2" : 0,
    "inReg3" : 0
}
```

読み込みのリクエストに対しては、全てのレジスタの情報を返します。

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		
7	request_from	リクエスト元クラウド ID	String		reply が true に設定されている時のみ表示
8	reply_to	リクエストメッセージの MD5 値	String		
9	result	取得ステータス	Boolean		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

■リクエストメッセージサンプル(レジスタの書き込み)

```
{
    “write”: {
        "inBit0" : 1,
        "inReg3" : 34586
    }
}
```

#	JSON キー	内容	データ型	必須	補足
1	write	書込み制御	JSON obj		モデリングファイルの設定に応じて異なります。

■応答メッセージサンプル

```
{
    "timestamp" : "2019-11-22T13:17:20.327+09:00",
    "unitId" : 255,
    "maker" : "PlatHome Co., Ltd.",
    "product" : "Virtual PLC server",
    "model" : "32 bits 8 port Register",
    "sku" : "0000",
    "request_from" : "0x01(0)",
    "reply_to" : "10348911fddda12f880575e5dec3d698",
    "result" : true,
    "inBit0" : 1,
    "inBit1" : 0,
    "inBit2" : 0,
    "inBit3" : 0,
    "inReg0" : 0,
    "inReg1" : 0,
    "inReg2" : 0,
    "inReg3" : 34586
}
```

書込みのリクエストに対しては、書き込み後の関連するレジスタセットの情報を返します。

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	unitId	Modbus Unit ID	Integer or String	○	WEB UI から設定された値 id_form キーの設定に応じて、整数・大文字 16 進数、小文字 16 進数の何れか.
3	maker	メーカー名	String		WEB UI から設定された値 モデリングの設定応じて非表示
4	product	プロダクト名	String		
5	model	モデル名	String		
6	sku	SKU 名	String		
7	request_from	リクエスト元クラウド ID	String		reply が true に設定されている時のみ表示
8	reply_to	リクエストメッセージの MD5 値	String		
9	result	取得ステータス	Boolean		

上記以外の項目は、モデリングファイルの設定に応じて異なります。

■リクエストメッセージサンプル(レジスタの書き込み)

```
{
    “write”: {
        "inBit10" : 1
    }
}
```

#	JSON キー	内容	データ型	必須	補足
1	write	書込み制御	JSON obj		モデリングファイルの設定に応じて異なります。

■応答メッセージサンプル(エラー出力)

```
{
    "timestamp" : "2019-11-22T13:38:15.589+09:00",
    "request_from" : "0x01(0)",
    "reply_to" : "8fe9b1439f2033966d8ed07bebcea539",
    "result" : false,
    "reason" : "parser_sub_socket(): invalid write request '{¥"inBitr10¥": 0}'"
}
```

#	JSON キー	内容	データ型	常駐	補足
1	timestamp	データ取得日時	String	○	ISO8601 拡張書式
2	request_from	リクエスト元クラウド ID	String	○	
3	reply_to	リクエストメッセージの MD5 値	String	○	
4	result	取得ステータス	String	○	成功時は true
5	reason	エラーの理由	String	○	

reply が true に設定されている時のみ、応答を返します。

複数の書き込みリクエストが、行われた場合で、そのいずれかの書き込みが成功した場合、エラー出力は行われず、正常に書き込まれたレジスタの情報のみが返されます。

PD Handler SW4x

PD Handler SW4x の出力メッセージは、モデリングファイルを定義することによりへ変更することができます。
本項では、PD Handler SW4x にハードコーディングされたデフォルトのメッセージフォーマットを記載します。
「常駐」項については、モデリングファイルで定義可能であるため記載いたしません。

6.3. SW42P0_1x01 温度センサーノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x00",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "temperature": 27.2,
    “tempeStatus”:true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x00
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	**経路情報**	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	temperature	**温度(度)** -79.9 ~ 79.9	Real	
7	tempeStatus	**温度の測定ステータス**	Boolean	

6.4. SW4210_1202 温・湿度センサーノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x01",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "temperature": 27.2,
    “humidity": 72.1,
    "tempeStatus":true,
    "humidStatus":true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x01
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	**経路情報**	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	temperature	**温度(度)** -79.9 ~ 79.9	Real	
7	humidity	**湿度**(%) 0.0 ~ 100.0	Real	
8	tempeStatus	**温度の測定ステータス**	Boolean	
9	humidStatus	**湿度の測定ステータス**	Boolean	

6.5. SW4210_1205 照度センサーノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x02",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "illuminance": 623,
    "illumStatus":true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x02
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	経路情報	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	illuminance	照度(lx) 0 ~ 99999	Integer	
7	illumStatus	照度の測定ステータス	Boolean	

6.6. SW4210_1204 温・湿度・照度センサーノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x03",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "temperature": 27.2,
    “humidity": 72.1,
    "illuminance": 623,
    "tempeStatus":true,
    "humidStatus":true,
    "illumStatus":true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x03
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	**経路情報**	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	temperature	**温度(度)** -79.9 ~ 79.9	Real	
7	humidity	**湿度**(%) 0.0 ~ 100.0	Real	
8	illuminance	**照度**(lx) 0 ~ 99999	Integer	
9	tempeStatus	**温度の測定ステータス**	Boolean	
10	humidStatus	**湿度の測定ステータス**	Boolean	
11	illumStatus	**照度の測定ステータス**	Boolean	

6.7. SW4220_1000 人感センサーノード（活動量）

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x09",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "humanCount": 451,
    “detectionMaxWidth": 200,
    "detectionMinWidth": 32
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x09
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	**経路情報**	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	humanCount	人感検出回数 0 ～ 4095	Integer	
7	detectionMaxWidth	人感検出幅 NAX 0 ～ 255	Integer	
8	detectionMinWidth	人感検出幅 MIN 0 ～ 255	Integer	

6.8. SW4240_1000 パルスカウントノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x0A",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "pulseCount1": 23567,
    “pulseCount2": 1327
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0A
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	経路情報	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	pulseCount1	パルスカウント数 1 0 ~ 99999999	Integer	
7	pulseCount2	パルスカウント数 2 0 ~ 99999999	Integer	
8	eepromStatus	EEPROM ステータス	Boolean	EEPROM R/W エラー発生時の出力

6.9. SW4220_1020 人感センサーノード（イベントドリブン仕様）

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x0B",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "humanDetectCount": 267,
    “periodic": true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0B
4	batteryVoltage	電池電圧の状態コード		
5	routes	経路情報	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	humanDetectCount	人感検知カウント 0 ~ 99999999	Integer	
7	periodic	定期送信を示すフラグ	Boolean	

6.10. SW42J0_1202 リモコン温湿度ノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x0D",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    "temperature": 27.2,
    “humidity": 72.1,
    "tempeStatus":true,
    "humidStatus":true
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0D
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	経路情報	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	temperature	温度(度) -79.9 ~ 79.9	Real	
7	humidity	湿度(%) 0.0 ~ 100.0	Real	
8	tempeStatus	温度の測定ステータス	Boolean	
9	humidStatus	湿度の測定ステータス	Boolean	
10	subCrtlCode	サブ制御コード	String	制御コマンドに対する応答
11	confData	設定データ．ノードの設定情報バイト列.	String	
12	cmdRespons	受信確認レスポンスバイト列	String	

6.11. SW42K0_1000 パルスピックノード

■データサンプル

```
{
    "timestamp": "2019-07-30T12:06:21.680+09:00",
    "unitId": "0xFA",
    "unitType": "0x0F",
    "batteryVoltage": "0x00",
    "routes": [
        {
            "unitId": "0x00",
            "RSSI": "0x2E"
        },
        {
            "unitId": "0xFA"
        }
    ],
    " integralPowerConsum ": 45678.9234
}
```

#	JSON キー	内容	データ型	補足
1	timestamp	データ取得日時	String	ISO8601 拡張書式
2	unitId	SID データの送信元 ID	String	
3	unitType	ユニットタイプコード	String	0x0F
4	batteryVoltage	電池電圧の状態コード	String	
5	routes	**経路情報**	JSON obj	unitId:中継ユニットの ID RSSI:経路情報の RSSI
6	integralPowerConsum	**積算電力(Kwh)** 0.0 ~ 99999999.9999	Real	
7	eepromStatus	EEPROM ステータス	Boolean	EEPROM R/W エラー発生時の出力

ぷらっとホーム株式会社
〒102-0073　東京都千代田区九段北 4-1-3　日本ビルディング九段別館 3F