Plat' Home
TECHNOLOGY to serve you.

OpenBlocks IoT Family向け
データーハンドリング設定
リファレンスガイド

Ver.3.4.2

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. 共通事項

1.1. モジュール

データハンドリングを実現する PD Repeater や PD Handler,PD Agent,PD Broker のアプリケーションを**モジュール**と呼びます。

1.2. 設定ファイル

1.2.1. フォーマット

各モジュールの設定ファイルは、JSON フォーマット記述さます。

1.2.2. デォルト値

各モジュールは、設定ファイルで設定可能なパラメタ（以下、**キー**と称します。）について、特に指定が無い限りデフォルト値を持ちます。

1.3. モジュール間通信

1.3.1. モジュール間通信メソッド

各モジュール間のデータ転送には、Unix ドメインソケットを用います。

1.3.2. ソケットのパス名

データを受け取るソケットのパス名はbindキーで、データの送り先ソケットのパス名はpush_toキーで変更できます。
ソケットのパス名の先頭が@の場合は、Abstract name spaceと解釈されます。

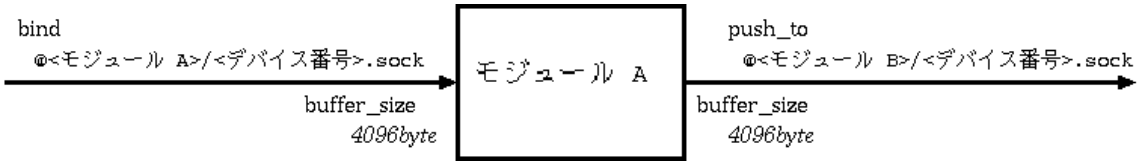
1.3.3. データーサイズ

データーのバッファサイズはデフォルト4,096 byteです。データを受け取るモジュールにおいてはbuffer size'キーで変更できます。

1.3.4. モジュール間通信に関わるモジュール共通のキー

キー	データ型	説明
bind	文字列	データを受け取るソケット名．文字列の先頭が'@' の場合はabstract namespace と解釈します． pd broker を除き空の場合は、各モジュールのデフォルト値が用いられます． @/<モジュール名＞/＜デバイス番号＞.sock の形式で指定します．
push_to	文字列	データの送り先ソケット名．文字列の先頭が'@' の場合はabstract namespace と解釈します． pd broker を除き空の場合は、各モジュールのデフォルト値が用いられます． @/<モジュール名＞/＜デバイス番号＞.sock の形式で指定します．
buffer_size	整数値	データのバッファサイズ(byte)

モジュール間通信に関わるモジュール共通のキー



2. PD Repeater

2.1. デフォルトパス

PD Repeater に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_repeater	常駐実行オブジェクト（デーモン）
/usr/sbin/pd_repeater_p	実行オブジェクト（ユーティリティー）
/lib/systemd/system/pd_repeater.service	Systemd Service ファイル
/etc/init.d/pd_repeater	RC ファイル
/var/webui/config/pd_repeater.conf	設定ファイル
/var/run/pd_repeater.pid	PID ファイル
/var/webui/pd-data/pd_repeater.db	キャッシュファイル
@/pd_repeater/＜デバイス番号＞.sock	入力 UNIX ドメインソケット
@/pd_handler/＜デバイス番号＞.sock	出力 UNIX ドメインソケット

PD Repeater に関連するファイルのデォルトパス

2.2. 設定ファイルの書式

2.2.1. 構文

```
{
    “db_file” : “<キャッシュファイルパス名>“,
    “max_db_size” : <キャッシュサイズ>,
    “servers”: {
        “<クラウドインデックスキー>“ : {
            <サーバ設定用JSON オブジェクト>
        },
        “<クラウドインデックスキー>“ : {
            <サーバの設定用JSON オブジェクト>
        }
      },
      “devices”: [
          {
              “localname” : “<デバイスのローカル名>“,
              “bind” : “<データを受け取るソケット名>“,
              “push_to” : “<データの送り先ソケット名>“,
              “buffer_size” : <データのバッファサイズ>,
              “truncate” : <入力データを受け付けない時間>,
              “receive” : <下流方向制御>,
              “<クラウドインデックスキー>“ : {
                  <クラウド固有のデバイス設定用JSON オブジェクト>
              },
              “<クラウドインデックスキー>“ : {
                  <クラウド固有のデバイス設定用JSON オブジェクト>
              }
          }
      ]
  }
```

2.2.2. ルートオブジェクト

キー	データ型	説明
db_file	文字列	キャシュファイルのパス名. デフォルト値は '/var/webui/pd-data/pd_repeater.db'. (MAXPATHLEN)
max_db_size	整数値	キャシュファイルの最大サイズ(Mbyte). デフォルト値は 16.
servers	JSON obj	servers オブジェクト
devices	JSON 配列	devices オブジェクト

ルートオブジェクト

2.2.3. servers オブジェクト

キー	データ型	説明
pd_ex	JSON obj	PD Exchenge へデータを送るための設定オブジェクト
mqtt	JSON obj	汎用の MQTT ブローカーへデータを送るための設定オブジェクト
w4d	JSON obj	IBM Watson IoT for device へデータを送るための設定オブジェクト
kinesis	JSON obj	Amazon Kinesis へデータを送るための設定オブジェクト
pgsql	JSON obj	PostgreSQL データベースへデータを送るための設定オブジェクト
mysql	JSON obj	MySQL データベースへデータを送るための設定オブジェクト
web	JSON obj	汎用の Web サーバへデータを送るための設定オブジェクト
awsiot	JSON obj	Amazon AWS IoT へデータを送るための設定オブジェクト
eventhub	JSON obj	Microsoft Azure Event Hubs へデータを送るための設定オブジェクト
w4g	JSON obj	IBM Watson IoT for gateway へデータを送るための設定オブジェクト
iothub	JSON obj	Microsoft Azure IoT Hub へデータを送るための設定オブジェクト
t4d	JSON obj	NTT docomo Toami for docomo へデータを送るための設定オブジェクト
lsocket	JSON obj	Unix ドメインソケットへデータを送るための設定オブジェクト
kddi_std	JSON obj	KDDI IoT クラウド Standard へデータを送るための設定オブジェクト
nf_dvhub	JSON obj	NIFTY IoT Device Hub へデータを送るための設定オブジェクト
pd_web	JSON obj	Plat'Home 独自仕様の Web サーバへデータを送るための設定オブジェクト
iotcore	JSON obj	Google Cloud IoT Core へデータを送るための設定オブジェクト
ltcp	JSON obj	TCP ソケットへデータを送るための設定オブジェクト
awsiot_ws	JSON obj	Amazon AWS IoT へ MQTT over WEbsocket でデータを送るための設定オブジェクト
iothub_ws	JSON obj	Microsoft Azure IoT Hub へ MQTTover Websocket でデータを送るための設定オブジェクト
dummy	JSON obj	ログ出力のみ行うダミー出力ための設定オブジェクト
sbiot	JSON obj	SoftBank IoT へデータを送るための設定オブジェクト
things	JSON obj	Things クラウドへデータを送るための設定オブジェクト
ftp	JSON obj	FTP サーバへデータを送るための設定オブジェクト

server オブジェクト

クラウドインデックスキーとして同文字列に'_0' 又は'_1', '_2', '_3'を付加した文字列(例えば'pd_ex' について、'pd_ex_0','pd_ex_1','pd_ex_2','pd_ex_3') を用いることで、同一のクラウドの異る 4 つのサーバーもしくはエンドポイントを設定するこのも可能です。

2.2.3.1. PD Excheng(pd_ex)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
url	文字列	接続先URL. デフォルト値は'http://pd.plathome.com'. (MAXPATHLEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse
poll_interval	整数値	PD Exchange からコマンド(下流方向へのメッセージ) を読み出す間隔(sec). デフォルト値は30.
deid_prefix	文字列	PD Exchange のデバイス ID プリフィック. (10byte)
secretkey	文字列	接続に用いる secretkey. (16byte)

PD Exchange (pd ex) の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.2. 汎用 MQTT ブローカー(mqtt)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホストをIP アドレス又はFQDN で指定. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は1883(TCP 接続時) 又は 8883(SSL 接続時).
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
client_id	文字列	MQTT 接続に用いる Client ID. (64byte)
topic_prefix	文字列	MQTT の送信トピックの先頭に付加する文字列. 送信トピックは、本文字列とデバイス設定オブジェクト'unique id' に指定される文字列の組み合わせで、*topic_prefix/unique_id* となる. (MAXPATHLEN)
rcv_topic_prefix	文字列	MQTT の受信トピックの先頭に付加する文字列. 受信トピックは、*rcv_topic_prefix/#* で待ち受け、デバイス設定オブジェクト 'unique_id' に指定される文字列の組み合わせからなる *rcv_topic_prefix/unique_id* との完全一致するデバイスへ受信メッセージを内部へ転送する. 完全一致しない場合は、前方一致により一致する複数のデバイスへ受信メッセージを内部へ転送する. (MAXPATHLEN)
lwt_topic_prefix	文字列	LWT の送信トピックの先頭に付加する文字列. 送信トピックは、本文字列とデバイス設定オブジェクト 'unique_id' に指定される文字列の組み合わせで、*lwt_topic_prefix/unique_id*となる.(MAXPATHLEN)
lwt_message	文字列	LWT の送信メッセージ. (128byte)
username	文字列	接続に用いるユーザー名. (32byte)
password	文字列	パスワード認証に用いるパスワード. (128byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

汎用 MQTT ブローカー(mqtt)の設定オブジェクト

2.2.3.3. IBM Watson IoT for device (w4d)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
domain	文字列	接続先ドメイン名. デフォルト値は 'messag-ing.internetofthings.ibmcloud.com'. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は1883(TCP 接続時) 又は 8883(SSL 接続時).
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
org_id	文字列	組織 ID. デフォルト値は'quickstart'. (16byte)
event_id	文字列	イベントID. デフォルト値は'sample'. (128byte)
lwt_event_id	文字列	LWT のイベント ID. デフォルト値は 'lwt'.
format_string	文字列	データーフォーマット. デフォルト値は'json'. (16byte)
lwt_message	文字列	LWT の送信メッセージ. (128byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)

IBM Watson IoT for device (w4d) の設定オブジェクト

2.2.3.4. Amazon Kinesis (kinesis)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
domain	文字列	接続先ドメイン名. デフォルト値は'amazonaws.com'. (MAXHOSTNAMELEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse
region	文字列	接続先リージョン名. デフォルト値は'ap-northeast-1'. (64byte)
accessid	文字列	アクセスID. (128byte)
accesskey	文字列	アクセスキー. (128byte)
streamname	文字列	ストリーム名. (128byte)

Amazon Kinesis (kinesis) の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.5. PostgreSQL (pgsql)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
hostaddr	文字列	接続先IP アドレス. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は5432.
sslmode	文字列	PostgreSQL の SSL モード. デフォルト値は'verify-full'. (12byte)
pg_type	整数値	postgresql/server/catalog/pg_type.h に定義されるOID. デフォルト値は25 (TEXTOID).
dbname	文字列	データベース名. デフォルト値は'pd repeater'. (32byte)
username	文字列	接続に用いるユーザー名. (32byte)
password	文字列	接続に持ちるパスワード. (32byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

PostgreSQL (pgsql) の設定オブジェクト

2.2.3.6. MySQL (mysql)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
hostaddr	文字列	接続先IP アドレス. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は3306.
dbname	文字列	データベース名. デフォルト値は'pd repeater'. (32byte)
username	文字列	接続に用いるユーザー名. (32byte)
password	文字列	接続に持ちるパスワード. (32byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

MySQL (mysql) の設定オブジェクト

2.2.3.7. 汎用の Web サーバ(web)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
url	文字列	接続先URL. (MAXPATHLEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse
username	文字列	BASIC 認証用ユーザー名. 空の場合、認証は行わない. (32byte)
password	文字列	BASIC 認証用パスワード. 空の場合、認証は行わない. (32byte)
max_post_msize	整数値	最大ポストサイズ(Mbyte). デフォルト値は 1.
content_type	文字列	'Content-Type' を指定する. 'text/plain' もしくは'application/json' が指定された場合はペイロードのURL セーフエンコードを行わない. デフォルト値は無指定. (64byte)

汎用の Web サーバ(web) の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.8. Amazon AWS IoT (awsiot)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホスト名. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は8883.
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
rootCA	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)

Amazon AWS IoT (awsiot) の設定オブジェクト

2.2.3.9. Microsoft Azure Event Hubs (eventhub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
domain	文字列	接続先ドメイン名. デフォルト値は'servicebus.windows.net'. (MAX-HOSTNAMELEN)
namespace	文字列	接続先名前空間. (64byte)
port	整数値	接続先ポート番号. デフォルト値は5671.

Microsoft Azure Event Hubs (eventhub)の設定オブジェクト

2.2.3.10. IBM Watson IoT for gateway (w4g)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
domain	文字列	接続先ドメイン名. デフォルト値は 'messag-ing.internetofthings.ibmcloud.com'. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
port	整数値	接続先ポート番号. デフォルト値は1883(TCP 接続時) 又は 8883(SSL 接続時).
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
org_id	文字列	組織 ID. デフォルト値は'quickstart'. (16byte)
gateway type	文字列	ゲートウェイタイプデフォルト値は'sample'. (20byte)
gateway id	文字列	ゲートウェイ ID. (20byte)
event_id	文字列	イベントID. デフォルト値は'sample'. (128byte)
lwt_event_id	文字列	LWT のイベント ID. デフォルト値は 'lwt'.
format_string	文字列	データーフォーマット. デフォルト値は'json'. (16byte)
lwt_message	文字列	LWT の送信メッセージ. (128byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

IBM Watson IoT for gateway (w4g) の設定オブジェクト

2.2.3.11. Microsoft Azure IoT Hub(iothub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
domain	文字列	接続先ドメイン名. デフォルト値は'azure-devices.net'. (MAXHOSTNAMELEN)
hub_name	文字列	接続先ハブ名. (64byte)
gw_host	文字列	ゲートウェイホスト名.(MAXHOSTNAMELEN)
tail_slash	論理値	MQTT トピックの末尾に'/'を付けるか否か. デフォルト値は true
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
lwt_property_bag	文字列	LWT Property Bag. (256byte)
lwt_message	文字列	LWT の送信メッセージ. (128byte)
rootCA	文字列	X.509 認証に用いる Root CA 証明書ファイルのパス名. (MAXHOSTNAMELEN)

Microsoft Azure IoT Hub(iothub)の設定オブジェクト

2.2.3.12. NTT docomo Toami for docomo (t4d)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
url	文字列	接続先URL. デフォルト値は'https://xxxx.to4do.com:443'. (MAX-PATHLEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse

NTT docomo Toami for docomo (t4d)の設定オブジェクト

2.2.3.13. UNIX ドメインソケット（lsocket）

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
root_path	文字列	送り先Unix ドメインソケットのルートパス名. 文字列の先頭が'@'の場合はabstract namespace と解釈する. デフォルト値は'/tmp'. (MAX-PATHLEN)

UNIX ドメインソケット（lsocket)の設定オブジェクト

2.2.3.14. KDDI IoT クラウド Standard (kddi std)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
domain	文字列	接続先ドメイン名. デフォルト値は 'datalink.m2m-cloud-std.kddi.ne.jp'. (MAXHOSTNAMELEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse
port	整数値	接続先ポート番号. デフォルト値は443.
termid	文字列	端末ID. (16byte)
username	文字列	BASIC 認証用ユーザー名. 空の場合、認証は行わない. (32byte)
password	文字列	BASIC 認証用パスワード. 空の場合、認証は行わない. (32byte)

KDDI IoT クラウド Standard (kddi std)の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.15. NIFTY IoT Device Hub (nf_dvhub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホスト名. デフォルト値は'iot-device.jp-east-1.mqtt.cloud.nifty.com' (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'tcp'.
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
rootCA	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)

NIFTY IoT Device Hub (nf_dvhub)の設定オブジェクト

NIFTY IoT Device Hub は、2018 年 11 月 30 日をもってサービスを終了しています。
また、NIFTY IoT Device Hub と DEVICEHUB(https://www.devicehub.net)は、異なるサービスであり、本設定オブジェクトを用いて DEVICEHUB にデータを送ることはできません。

2.2.3.16. Plat'Home 独自仕様 Web サーバ(pd web)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
url	文字列	接続先URL. (MAXPATHLEN)
ssl_verify	論理値	サーバー証明書の検証を行うか否か. デフォルト値はfalse
poll_interval	整数値	PD Web ではメッセージの送信と同時に下流方向へのメッセージを取得するが、poll interval には送信すべきメッセージが存在しない場合を想定し、空の接続を起こす間隔(sec) を指定する. デフォルト値は 30.
username	文字列	BASIC 認証用ユーザー名. 空の場合、認証は行わない. (32byte)
password	文字列	BASIC 認証用パスワード. 空の場合、認証は行わない. (32byte)
max_post_msize	整数値	最大ポストサイズ(Mbyte). デフォルト値は 1.

Plat'Home 独自仕様 Web サーバ(pd web)の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.17. Google Cloud IoT Core (iotcore)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホスト名(MAXHOSTNAMELEN)
port	整数値	接続先ポート番号. デフォルト値は8883.
project_id	文字列	プロジェクトID. (32bytes)
cloud_region	文字列	Cloud リージョン. (16bytes)
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
lwt_message	文字列	LWT の送信メッセージ. (128byte)
rootCA	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)

Google Cloud IoT Core (iotcore)の設定オブジェクト

2.2.3.18. TCP ソケット (ltcp)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
ip_addr	文字列	接続先IP アドレス. デフォルト値は'127.0.0.1'. (16byte)
delimiter	文字列	メッセージのセパレータコード. デフォルト値は'0x00'. 例えばCRLF コードをセパレータとする場合は'0x0d0a'. (7byte)

TCP ソケット (ltcp)の設定オブジェクト

2.2.3.19. Amazon AWS IoT [Websocket](awsiot_ws)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホスト名. (MAXHOSTNAMELEN)
port	整数値	接続先ポート番号. デフォルト値は8883.
keepaliveinterval	整数値	MQTT のKeepalive 間隔(sec). デフォルト値は10.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
accessid	文字列	アクセス ID. (128byte)
accesskey	文字列	アクセスキー. (128byte)

Amazon AWS IoT[Websocket](awsiot_ws) の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.20. Microsoft Azure IoT Hub[Websocket] (iothub_ws)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
domain	文字列	接続先ドメイン名. デフォルト値は'azure-devices.net'. (MAXHOSTNAMELEN)
hub_name	文字列	接続先ハブ名. (64byte)
keepaliveinterval	整数値	MQTT のKeepalive 間隔(sec). デフォルト値は10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_retained	論理値	LWT の Retained. デフォルト値は false.
lwt_property_bag	文字列	LWT Property Bag. (256byte)
lwt_message	文字列	LWT の送信メッセージ. (128byte)

Microsoft Azure IoT Hub[Websocket](iothub_ws)の設定オブジェクト

Proxy を使用する場合は、環境変数 http_proxy 又は https_proxy に設定します。

2.2.3.21. ダミー出力（dummy）

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
log	論理値	ペイロードとそのサイズのログ出力を行う. デフォルト値は false.

ダミー出力(dummy)の設定オブジェクト

2.2.3.22. SoftBank IoT (sbiot)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホストをIP アドレス又はFQDN で指定. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'ssl'.
port	整数値	接続先ポート番号. デフォルト値は1883(TCP 接続時) 又は 8883(SSL 接続時).
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
retained	論理値	MQTT の Retained. デフォルト値は false.
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
resp	論理値	応答出力を受け取るか否か. デフォルト値は false.
cloud_id	文字列	MQTT の送信トピックに与えられる送信先クラウド名. デフォルト値は'CSE1000'.(64byte)
client_id	文字列	MQTT 接続に用いる Client ID. (96byte)
ae_id	文字列	接続に用いる SoftBank IoT の AE-ID.(64byte)
password	文字列	パスワード認証に用いるパスワード. (128byte)
asset	文字列	接続に用いる SoftBank IoT のアセット名.(64byte)
truststore	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. (MAXPATHLEN)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

SoftBank IoT (sbiot) の設定オブジェクト

SoftBank IoT に関わる実装は、SoftBank IoT プラットフォームネットワーク接続仕様(Rev.1.1.1)に準拠し「SoftBank スマ可視化専用クラウド」の仕様に合わせて実装されています。

2.2.3.23. Things クラウド(things)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
res_memory	整数値	常駐メモリの増加分の上限値(Mbyte). 上限値を超えると送信プロセスを再起動する. デフォルト値は32.
host	文字列	接続先ホストをIP アドレス又はFQDN で指定. (MAXHOSTNAMELEN)
protocol	文字列	接続プロトコル'tcp' 又は'ssl' を指定. デフォルト値は'ssl'.
port	整数値	接続先ポート番号. デフォルト値は1883(TCP 接続時) 又は 8883(SSL 接続時).
keepaliveinterval	整数値	MQTT の Keepalive 間隔(sec). デフォルト値は 10.
cleansession	論理値	MQTT の Clean session. デフォルト値は true.
reliable	論理値	MQTT の Reliable. デフォルト値は true.
qos	整数値	MQTT の送信用 QoS, 0～2 を指定. デフォルト値は 0.
topic	文字列	MQTT の送信トピック. デフォルト値は'measurement/mesurements/create'. (MAXPATHLEN)
rcv_qos	整数値	MQTT の受信用 QoS, 0～2 を指定. デフォルト値は 1.
rcv_topic	文字列	MQTT の受信トピックの先頭に付加する文字列. 受信トピックは、デバイス設定オブジェクト'client_id' に指定される文字列と組み合わせ rcv_topic_prefix/client_id となる. (MAXPATHLEN)
lwt	論理値	Last Will and Testament (LWT) を使用するか否か. デフォルト値は false.
lwt_qos	整数値	LWT の QoS, 0～2 を指定. デフォルト値は 0.
lwt_message	文字列	LWT の送信メッセージ. (128byte)
rootCA	文字列	SSL 接続に用いる Root CA 証明書ファイルのパス名. デフォルト値は'/etc/ssl/certs/ca-certificates.crt'.(MAXPATHLEN)
tenant_id	文字列	パスワード認証に用いるユーザー名の先頭に付与する文字列. デフォルト値は'iotconnect'.(64byte)
username	文字列	パスワード認証に用いるユーザー名のサフックス. ユーザー名は'tenant_id'に指定される文字列と組み合わせ tenant_id/username となる.(64byte)
password	文字列	パスワード認証に用いるパスワード. (128byte)

Things クラウド(things)の設定オブジェクト

Things クラウドに関わる実装は、2020 年 6 月現在試験的なもので、設定オブェクトについては予告なく変更される場合があります。

2.2.3.24. FTP サーバ(ftp)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
interval	整数値	サーバへの送信間隔(sec). デフォルト値は 30.
expier	整数値	キャッシュのデータ保持時間(sec). デフォルト値0 の場合はデータが送信さるまで無制限.
restart	整数値	送信プロセスの再起動間隔(sec). デフォルト値は604800、0 の場合は再起動しない.
url	文字列	'ftp://' から始まる接続先 URL. (MAXPATHLEN)
username	文字列	接続に用いるユーザー名. (32byte)
password	文字列	パスワード認証に用いるパスワード. (32byte)
per_device	論理値	デバイス毎にファイル名を変えて送信する. デフォルト値は false.
fn_prefix	文字列	ファイル名のプリフィック. per_device キーが false の時のみ有効. (64byte).
fn_suffix	文字列	ファイル名のサフィック. per_device キーが false の時のみ有効. (64byte).
fn_time	文字列	ファイル名のプリフィックとサフィックの間に挿入される PD Repeater がデータを受け取った時刻文字列のの仕様. 'none'(挿入無し), 'oldest'(最古), 'latest'(最新), 'both'(両方) のいずれか. 'oldest' 又は 'latest' 場合は 'CCyymmddHHMMSS'、 'both' の場合は、'CCyymmddHHMMSS-CCyymmddHHMMSS'.

FTP サーバ(ftp)の設定オブジェクト

FTP サーバに関わる実装は、2020 年 6 月現在試験的なもので、設定オブェクトについては予告なく変更される場合があります。

2.2.4. device オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
bind	文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/＜デバイス番号＞.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte). デフォルト値は4096
truncate	実数値	データを間引くための入力データを受け付けない時間(msec). デフォルト値0 の場合は間引かない.
receive	論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse
pd_ex	JSON obj	PD Exchenge 固有の設定オブジェクト
mqtt	JSON obj	汎用の MQTT ブローカー固有の設定オブジェクト
w4d	JSON obj	IBM Watson IoT for device 固有の設定オブジェクト
kinesis	JSON obj	Amazon Kinesis 固有の設定オブジェクト
pgsql	JSON obj	PostgreSQL データベース固有の設定オブジェクト
mysql	JSON obj	MySQL データベース固有の設定オブジェクト
web	JSON obj	汎用の Web サーバ固有の設定オブジェクト
awsiot	JSON obj	Amazon AWS IoT 固有の設定オブジェクト
eventhub	JSON obj	Microsoft Azure Event Hubs 固有の設定オブジェクト
w4g	JSON obj	IBM Watson IoT for gateway 固有の設定オブジェクト
iothub	JSON obj	Microsoft Azure IoT Hub 固有の設定オブジェクト
t4d	JSON obj	NTT docomo Toami for docomo 固有の設定オブジェクト
lsocket	JSON obj	Unix ドメインソケットへデータを送るための固有の設定オブジェクト
kddi_std	JSON obj	KDDI IoT クラウド Standard 固有の設定オブジェクト
nf_dvhub	JSON obj	NIFTY IoT Device Hub 固有の設定オブジェクト
pd_web	JSON obj	Plat'Home 独自仕様の Web サーバ固有の設定オブジェクト
iotcore	JSON obj	Google Cloud IoT Core 固有の設定オブジェクト
ltcp	JSON obj	TCP ソケット固有の設定オブジェクト
awsiot_ws	JSON obj	Amazon AWS IoT へ MQTT over Websocket でデータを送るための固有の設定オブジェクト
iothub_ws	JSON obj	Microsoft Azure IoT Hub へ MQTT over Websocket でデータを送るための固有の設定オブジェクト
dummy	JSON obj	ダミー出力 固有の設定オブジェクト
sbiot	JSON obj	SoftBank IoT へデータを送るための固有の設定オブジェクト
things	JSON obj	Things クラウドへデータを送るための固有の設定オブジェクト
ftp	JSON obj	FTP サーバへデータを送るための固有の設定オブジェクト

devices オブジェクト

クラウドインデックスキーについては servers オブジェクトと同様に同文字列に'_0' 又は '_1', '_2', '_3' を付加した文字列(例えば'pd_ex' について、'pd_ex_0','pd_ex_1','pd_ex_2', 'pd_ex_3) を用いることが可能です。

2.2.4.1. PD Exchenge(pd_ex)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
deid suffix	文字列	PD Exchange のデバイス ID サフックス. (9byte)

PD Exchenge(pd_ex)固有の設定オブジェクト

2.2.4.2. 汎用の MQTT ブローカー(mqtt)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
unique_id	文字列	ユニーク ID. (20byte)

汎用の MQTT ブローカー(mqtt)固有の設定オブジェクト

2.2.4.3. IBM Watson IoT for device (w4d)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
device_id	文字列	デバイス ID. (20byte)
device_type	文字列	デバイスタイプ. (20byte)
password	文字列	パスワード. (128byte)
keystore	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

IBM Watson IoT for device (w4d)固有の設定オブジェクト

2.2.4.4. Amazon Kinesis (kinesis)

キー	データ型	説明
enable	論理値	デフォルト値はfalse

Amazon Kinesis (kinesis)固有の設定オブジェクト

2.2.4.5. PostgreSQL (pgsql)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
table	文字列	データを書き込むテーブル名. デフォルト値は 'buf'.
colum	文字列	データを書き込むカラム名. デフォルト値は 'jsonObj'.

PostgreSQL (pgsql)固有の設定オブジェクト

2.2.4.6. MySQL (mysql)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
table	文字列	データを書き込むテーブル名. デフォルト値は 'buf'.
colum	文字列	データを書き込むカラム名. デフォルト値は 'jsonObj'.

MySQL (mysql)固有の設定オブジェクト

2.2.4.7. 汎用の Web サーバ(web)

キー	データ型	説明
enable	論理値	デフォルト値はfalse

汎用の Web サーバ(web)固有の設定オブジェクト

2.2.4.8. Amazon AWS IoT (awsiot)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
client_id	文字列	クライアント ID. (20byte)
thing_name	文字列	Device Shadows に用いる thingName. (128byte)
topic	文字列	MQTT の送信トピック. (MAXPATHLEN)
rcv_topic	文字列	MQTT の受信トピック. (MAXPATHLEN)
lwt_topic	文字列	LWT の送信トピック. (MAXPATHLEN)
lwt_message	文字列	LWT の送信メッセージ. (128byte)
cert	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

Amazon AWS IoT (awsiot)固有の設定オブジェクト

2.2.4.9. Microsoft Azure Event Hubs (eventhub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
hub_name	文字列	Event Hubs 名. (64byte)
sas_policy	文字列	SAS ポリシー. (64byte)
sas_key	文字列	SAS キー. (64byte)

Microsoft Azure Event Hubs (eventhub)固有の設定オブジェクト

2.2.4.10. IBM Watson IoT for gateway (w4g)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
device_id	文字列	デバイス ID. (20byte)
device_type	文字列	デバイスタイプ. (20byte)

IBM Watson IoT for gateway (w4g)固有の設定オブジェクト

2.2.4.11. Microsoft Azure IoT Hub(iothub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
device_id	文字列	デバイス ID. (128byte)
device_key	文字列	デバイスキー. (64byte)
transparent	論理値	IoT Edge を透過モードで動作させるか否か. デフォルト値は false
module_id	文字列	モジュール ID. (64byte). 透過モード時は無視.
cert	文字列	X.509 認証に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	X.509 認証に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

Microsoft Azure IoT Hub(iothub)固有の設定オブジェクト

2.2.4.12. NTT docomo Toami for docomo (t4d)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
gwname	文字列	GatewayName. (32byte)
appkey	文字列	AppKey. (64byte)

NTT docomo Toami for docomo (t4d)固有の設定オブジェクト

2.2.4.13. UNIX ドメインソケット (lsocket)

キー	データ型	説明
enable	論理値	デフォルト値はfalse

UNIX ドメインソケット (lsocket)固有の設定オブジェクト

2.2.4.14. KDDI IoT クラウド Standard (kddi std)

キー	データ型	説明
enable	論理値	デフォルト値はfalse

KDDI IoT クラウド Standard (kddi std)固有の設定オブジェクト

2.2.4.15. NIFTY IoT Device Hub (nf_dvhub)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
event_type	文字列	イベントタイプ. (20byte)
device_id	文字列	デバイス ID. (20byte)
device_api_key	文字列	デバイス API キー(169byte)

NIFTY IoT Device Hub (nf_dvhub)固有の設定オブジェクト

NIFTY IoT Device Hub は、2018 年 11 月 30 日をもってサービスを終了しています。
また、NIFTY IoT Device Hub と DEVICEHUB(https://www.devicehub.net)は、異なるサービスであり、本設定オブジェクを用いて DEVICEHUB にデータを送ることはできません。

2.2.4.16. Plat'Home 独自仕様 Web サーバ(pd web)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
id	文字列	クライアント ID. (128byte)
key	文字列	トークン認証に用いる文字列(アクセスキー). (128byte)

Plat'Home 独自仕様 Web サーバ(pd web)固有の設定オブジェクト

2.2.4.17. Google Cloud IoT Core (iotcore)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
registry_id	文字列	レジストリ ID. (32byte)
device_id	文字列	デバイス ID. (32byte)
jwt_algorithm	文字列	JWT アルゴリズム. 'RS256', 'ES256' のいずれか.(6byte)
cert	文字列	SSL 接続に用いる証明書ファイルのパス名. (MAXPATHLEN)
privatekey	文字列	SSL 接続に用いる秘密鍵ファイルのパス名. (MAXPATHLEN)

Google Cloud IoT Core (iotcore)固有の設定オブジェクト

2.2.4.18. TCP ソケット (ltcp)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
port	文字列	接続先ポート番号. デフォルト値は 49152(TCP 接続時).

TCP ソケット (ltcp)固有の設定オブジェクト

2.2.4.19. Amazon AWS IoT[Websocket](awsiot_ws)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
client_id	文字列	クライアント ID. (20byte)
thing_name	文字列	Device Shadows に用いる thingName. (128byte)
topic	文字列	MQTT の送信トピック. (MAXPATHLEN)
rcv_topic	文字列	MQTT の受信トピック. (MAXPATHLEN)
lwt_topic	文字列	LWT の送信トピック. (MAXPATHLEN)
lwt_message	文字列	LWT の送信メッセージ. (128byte)

Amazon AWS IoT[Websocket](awsiot_ws)固有の設定オブジェクト

2.2.4.20. Microsoft Azure IoT Hub[Websocket] (iothub_ws)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
device_id	文字列	デバイス ID. (128byte)
device_key	文字列	デバイスキー. (64byte)

Microsoft Azure IoT Hub[Websocket](iothub_ws)固有の設定オブジェクト

2.2.4.21. ダミー出力(dummy)

キー	データ型	説明
enable	論理値	デフォルト値はfalse

ダミー出力(dummy)固有の設定オブジェクト

2.2.4.22. SoftBank IoT(sbiot)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
container	文字列	SoftBank IoT への接続に用いるコンテナ名. (64byte)

SoftBank IoT(sbiot)固有の設定オブジェクト

SoftBank IoT に関わる実装は、SoftBank IoT プラットフォームネットワーク接続仕様(Rev.1.1.1)に準拠し「SoftBank スマ可視専用クラウド」の仕様に合わせて実装されています。

2.2.4.23. Things クラウド(things)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
client_id	文字列	クライアント ID. (96byte)

Things クラウド(things)固有の設定オブジェクト

Things クラウドに関わる実装は、2020 年 6 月現在試験的なもので、設定オブェクトについては予告なく変更される場合があります。

2.2.4.24. FTP サーバ(ftp)

キー	データ型	説明
enable	論理値	デフォルト値はfalse
fn_prefix	文字列	ファイル名のプリフィック. per_device キーが true の時のみ有効. (64byte).
fn_suffix	文字列	ファイル名のサフィック. per_device キーが true の時のみ有効. (64byte).
fn_time	文字列	ファイル名のプリフィックとサフィックの間に挿入される PD Repeater がデータを受け取った時刻文字列のの仕様. 'none'(挿入無し), 'oldest'(最古), 'latest'(最新), 'both'(両方) のいずれか. 'oldest' 又は 'latest' 場合は 'CCyymmddHHMMSS'、 'both' の場合は、 'CCyymmddHHMMSS-CCyymmddHHMMSS'.

FTP サーバ(ftp)固有の設定オブジェクト

FTP サーバに関わる実装は、2020 年 6 月現在試験的なもので、設定オブェクトについては予告なく変更される場合があります。

2.3. 下流方向メッセージ

PD Repeater が下流のモジュールに転送するメッセージのフォーマットを示します。

Cloud ID (1Byte)	Sub ID (1Byte)	Header Size (2Bytes)	MD5 (16Bytes)	Header	Payload

フィールド	バイト数	説明
Cloud ID	1	便宜上割り当てている送受信先プロトコル（クラウド）の番号
Sub ID	1	便宜上割り当てている送受信先のサブ番号(0x00 又は 0x01)
Header Size	2	PD Reperater で付加された Header のサイズ
MD5	16	クラウドから送られて来た制御メッセージのハッシュ値(MD5)
Header	可変	PD Reperater で付加された MQTT のトピックや HTTP の応答ヘッダ
Payload	可変	クラウドから送られて来た制御メッセージ

下流方向メッセージのフォーマット

Cloud ID とヘッダーの内容を示します。

クラウド	Cloud ID	下流方向制御	ヘッダーの内容
PD Exchange	0x00	対応	Command ID, ApplicationID
MQTT サーバー	0x01	対応	MQTT 受信トピック
Watson IoT for Device	0x02	対応	MQTT 受信トピック
Amazon Kinesis	0x03		
PostgreSQL サーバ	0x04		
MySQL サーバ	0x05		
WEB サーバー	0x06		
AWS IoT	0x07	対応	MQTT 受信トピック
MS Azure Event hubs	0x08		
Watson IoT for Gateway	0x09	対応	MQTT 受信トピック
MS Azure IoT Hub	0x0a	対応	MQTT 受信トピック
Toami for DOCOMO	0x0b		
ドメインソケット	0x0c		
KDDI IoT クラウド Standard	0x0d		
IoT デバイスハブ	0x0e	対応	MQTT 受信トピック
Plat'Home 独自仕様 WEB サーバ	0x0f	対応	HTTP 応答ヘッダー
Google IoT Core	0x10	対応	MQTT 受信トピック
TCP	0x11	対応	接続先 IP アドレスとポート番号
AWS IoT [Websocket]	0x12	対応	MQTT 受信トピック
MS Azure IoT Hub[Websocket]	0x13	対応	MQTT 受信トピック
ダミー出力	0x14		
SoftBank IoT	0x15	対応	MQTT 受信トピックと oneM2M の JSON 制御用オブジェクト
Things クラウド	0x16	対応	MQTT 受信トピック
FTP サーバ	0x17		

クラウド ID とヘッダの内容

Watson IoT については、ペイロードを"d"キーの JSON オブジェクトとしているため、ハッシュ値（MD5）を得た後、デコードし"d"キーの JSON オブジェクトのみを制御メッセージとして出力しています。

IoT デバイスハブについては、ペイロードを"parameters"キーの JSON オブジェクトとしているため、ハッシュ値（MD5）を得た後、デコードし"parameters"キーの JSON オブジェクトのみを制御メッセージとして出力しています。

TCP の接続先 IP アドレスとポート番号は、次の JSON 文字列で付加されます。

{"ip_addr":"<*IP address*>","port":<*port*>}

SoftBank IoT については、oneM2M 仕様のペイロードから 'con' キーの JSON オブジェクトのみを出力し、これを除く JSON オブジェクトを MQTT 受信トピック文字列に連結しヘッダーとしています。

2.4. Plat'Home 独自仕様 WEB サーバ

2.4.1. HTTP ヘッダー

HTTP ヘッダー	内容
X-Pd-Web-Version	PD Web の仕様のバージョン番号
X-Pd-Web-Id	クライアントの ID
X-Pd-Web-Time	RFC3339 準拠のタイムスタンプ
X-Pd-Web-Md5	ペイロードのハッシュ値
X-Pd-Web-Signature	ヘッダ情報と鍵(Key)から作成されたハッシュ値
Content-Type	application/json;charset=UTF-8

2.4.2. トークン

PD Repeater で作成されるリクエストヘッダのトークン(X-Pd-Web-Sinature)は、Web サーバーにおいて次の PHP スクリプトにより再生できます。

```
$hash_hmac_data =
$_SERVER['HTTP_X_PD_WEB_VERSION'] .
$_SERVER['HTTP_X_PD_WEB_ID'] .
$_SERVER['HTTP_X_PD_WEB_TIME'] .
$_SERVER['HTTP_X_PD_WEB_MD5'];
$signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
```

ここで$keyは、は$_SERVER['HTTP X PD WEB ID'] と対をなす予めWebサーバーに保存された鍵(Key)となります。再生した$signature と$_SERVER['HTTP X PD WEB SIGNATURE'] を比較することで認証します。

応答ヘッダーのトークン(X-Pd-Web-Sinature)は、Web サーバーにおいて次の PHP スクリプトにより作成します。

```
$tm = localtime();
$timestamp = sprintf("%04d-%02d-%02dT%02d:%02d:%02d.000+09:00",
$tm[5]+1900,$tm[4]+1,$tm[3],$tm[2],$tm[1],$tm[0]);
$hash_hmac_data = '1.0' . $_SERVER['HTTP_X_PD_WEB_ID'] . $timestamp .
md5($payload) . $_SERVER['HTTP_X_PD_WEB_SIGNATURE'];
$signature = hash_hmac ('sha256', $hash_hmac_data, $key, false);
```

ここで、$payload は、ペイロードの文字列、送信すべき文字列（下流方向の制御メッセージ）が無い場合は、$payload="“; とします。
リクエストヘッダのトークンとは異なり、被署名文字列に PD Repeater から送られて来たリクエストヘッダのトークン($_SERVER['HTTP X PD WEB SIGNATURE'])が、含まれる点に注意して下さい。

3. PD Broker

3.1. デフォルトパス

PD Broker に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_broker	常駐実行オブジェクト（デーモン）
/lib/systemd/system/pd_broker.service	Systemd Service ファイル
/etc/init.d/pd_broker	RC ファイル
/var/webui/config/pd_broker.conf	設定ファイル
/var/run/pd_broker.pid	PID ファイル

PD Broker に関連するファイルのデォルトパス

3.2. 設定ファイルの書式

3.2.1. 構文

```
{
    “brokers” : [
        {
            “enable” : <bool値>,
            “bind”: “<受信ソケット名>“,
            “buffer_size” : <バッファサイズ>,
            “topic_size” : <トピックサイズ>,
            “destinations”:[
                {
                    “destination”: “<送信先ソケット名>“,
                    “topic”:”<トピックフィルタリング文字列>”,
                    “key”:”<キーフィルタリング文字列>”
                },
                {
                    “destination”: “<送信先ソケット名>“,
                    “topic”:”<トピックフィルタリング文字列>”,
                    “key”:”<キーフィルタリング文字列>”
                },

            ]
        },
        {
            “enable” : <bool値>,
            “bind”: “<受信ソケット名>“,
            “buffer_size” : <バッファサイズ>,
            “topic_size” : <トピックサイズ>,
            “destinations”:[
                {
                    “destination”: “<送信先ソケット名>“,
                    “topic”:”<トピックフィルタリング文字列>”,
                    “key”:”<キーフィルタリング文字列>”
```

```
                    },
                    {
                        “destination”: “<送信先ソケット名>“,
                        “topic”:”<トピックフィルタリング文字列>”,
                        “key”:”<キーフィルタリング文字列>”
                    },

                ]

            }
        ]
    }
```

3.2.2. brokers オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値はfalse
bind	文字列	データを受け取るソケット名．文字列の先頭が'@' の場合はabstract namespace と解釈します． デフォルト値は定義されていません。 @/<モジュール名>/<デバイス番号>.sock の形式で指定します.
buffer_size	整数値	データのバッファサイズ(byte)
topic_size	整数値	フィルタリングに用いるトピックのバッファサイズ(byte) デフォルト値は 256byte.
destinations	JSON 配列	destinations オブジェクト．最大 32 個まで指定可能

brokers オブジェクト

3.2.3. destinations オブジェクト

キー	データ型	説明
destination	文字列	データの送り先ソケット名．文字列の先頭が'@' の場合はabstract names-pace と解釈する．'@/*module name*/*localname*.sock' の形式で指定する.(MAXPATHLEN) デフォルト値は定義されていない．最大32 個まで指定可能
topic	文字列	トピックフィルタリングに用いる比較文字列. (topic_size に指定されるバイト数)
key	文字列	キーフィルタリングに用いる比較文字列. 64byte.

4. PD Agent

4.1. デフォルトパス

PD Agent に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_agent	常駐実行オブジェクト（デーモン）
/lib/systemd/system/pd_agent.service	Systemd Service ファイル
/etc/init.d/pd_agent	RC ファイル
/var/webui/config/pd_agent.conf	設定ファイル
/var/run/pd_agent.pid	PID ファイル

PD Agent に関連するファイルのデォルトパス

4.2. 設定ファイルの書式

4.2.1. 構文

```
{
    “agents” : [
        {
            “enable” : <bool値>,
            “localname:”<デバイス番号>“,
            “bind”: “<受信ソケット名>“,
            “push_to”: “<送信先ソケット名>“,
            “buffer_size” : <バッファサイズ>,
            “channels”:[
                {
                    <channelsオブジェクト>
                },
                {
                    <channelsオブジェクト>
                }
            ]
        },
        {
            “enable” : <bool値>,
            “localname:”<デバイス番号>“,
            “bind”: “<受信ソケット名>“,
            “push_to”: “<送信先ソケット名>“,
            “buffer_size” : <バッファサイズ>,
            “channels”:[
                {
                    <channelsオブジェクト>
                }
            ]
        }
    ]
}
```

4.2.2. agents オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値はfalse
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
bind	文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte). デフォルト値は4096
channels	JSON 配列	channels オブジェクト. 最大32 個まで設定可能.

agents オブジェクト

4.2.3. channels オブジェクト

キー	データ型	説明
name	文字列	チャンネルの名称. (32byte)
reply	論理値	'push to' キーに設定されるソケットに応答を返すか否かの設定. デフォルト値はtrue
dynamic_link	論理値	'exec' キーに指定されるパス名を Dynamic Link モジュールとして動作します. デフォルト値は false.
index	文字列	'exec' キーに指定される実行オブェクトの起動条件として評価される、JSON 文字列データのキー(32byte)
value	文字列	'exec' キーに指定される実行オブェクトの起動条件として評価される、JSON 文字列データの値(32byte)
exec	文字列	'index' キーに設定されるキーと'value' キーに設定されるその値が、JSON文字列データに含まれている場合に実行される実行オブェクトもしくは Dynamic Link モジュール又はLua言語スクリプトのパス名.(MAXPATHLEN). Dynamic Link モジュールの場合はパス名のディスクリプタに".so"を、Lua言語スクリプトの場合は".lua"を指定する.
args	文字列	'exec' キーに設定される実行オブェクトに与える引数. 'dynamic_link' キーが true の場合は無視されます.(1024byte)
lua_func	文字列	実行される Lua 言語スクリプト内の関数名.

channels オブジェクト

4.3. 実行オブジェクトに継承される環境変数

受信データのJSON 文字列の内、起動条件の評価に用いたキーと値を含め、値が文字列か数値の場合は、これを環境変数に設定し実行オブジェクトに継承します。
また、pd repeater から渡されたCloud ID、受信データ(ペイロード) のMD5 ハッシュ値、受信データのヘッダ情報とpd agent に設定されているデバイスのローカル名も合わせて環境変数に継承します。

環境変数名	データ型	説明
request_cloud_id	整数値	PD Repeater が便宜上割り当てている番号
request_sub_id	整数値	同一クラウドの複数サーバを利用している場合の識別子
request_header	文字列	PD Repeater から渡される受信データのヘッダ情報
request_payload	文字列	PD Repeater から渡される受信データ(ペイロード)
request_md5	文字列	PD Repeater から渡される受信データ(ペイロード) のMD5 ハッシュ値
agent_localname	文字列	'localname' キーに設定されているデバイスのローカル名(デバイス番号).
agent_bind	文字列	'bind' キーに設定されているデータを受け取ったソケット名.
agent_push_to	文字列	'push to' キーに設定されている応答先ソケット名.
agent_buffer_size	整数値	'buffer size' キーに設定されているデータのバッファサイズ.(byte)

実行オブジェクトに環境変数として継承されるペイロード以外のパラメータ

4.4. Dynamic Link モジュール

Dynamic Link モジュールのサンプルコードを示します。
Dynamic Link モジュールは、dlopen()によって PD Agent に取り込まれます。dl_exec() が、execv() によって実行されう外部の実行オブジェクトの代わりに呼び出される関数です。dl_exec() の引数には、実行オブジェクトを実行する際に環境変数として継承される全てのパラメータを含みます。　文字列ポインタ result が NULL の場合、応答メッセージの‘result’キーの値は ‘done’となります。　dl_init() は PD Agent の起動時に、dl_fini() は停止時に一度だけ呼ばれる関数であり、必要が無ければサンプルコードのままとして下さい。
この、サンプルコードは、/usr/share/phhms/agent にインストールされています。

```
/*
 *   Copyright (c) 2018
 *          Plat'Home CO., LTD. <support@plathome.co.jp>. All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *      notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *      notice, this list of conditions and the following disclaimer in the
 *      documentation and/or other materials provided with the distribution.
 * 3. Neither the name of the Plat'Home CO., LTD. nor the names of
 *      its contributors may be used to endorse or promote products derived
```

```
 *      from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.    IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <syslog.h>
#include <time.h>
#include <syslog.h>
#include <time.h>
#include <unistd.h>
#include <errno.h>
#include <sys/param.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <jansson.h>

#define N_CHANNEL           32

struct channel_t {
        unsigned char   reply;
        unsigned char   dynamic_link;
        char            name[32];               /* Name of chaanel                  */
        char            index[32];              /* Index of JSON string to be driven.   */
        char            value[32];              /* Value of JSON string to be driven.   */
        char            exec[MAXPATHLEN];   /* Pathname of dynamic link module.     */
        int             argc;                   /* Not use in dynamic link module.      */
        char            *argv[16];              /* Not use in dynamic link module.      */
        void            *dl_handle;             /* Handle on the global symbol object.  */
};

struct agent_t {
        int             num_ch;                 /* Total number of channels in device.  */
        unsigned char   enable;
        char            localname[32];          /* Local Device name (number).          */
        char            bind[MAXPATHLEN];   /* Pathname of UNIX domain socket to listen.    */
        size_t          buffer_size;            /* Buffer size of UNIX domain socket.   */
        char            push_to[MAXPATHLEN];    /* Pathname of UNIX domain socket to send. */
        struct channel_t *ch[N_CHANNEL];        /* Pointer to each channel.             */
        char            *buf;                   /* message buffer for down stream.      */
};

// Your own functions

/*
        Your own initialize function : void dl_init()
          It is called only once at startup.
```

```
        Return value : none.
*/
void dl_init()
{
        return;
}

/*
        Your own finalize function : void dl_fini()
          It is called only once at termination.
          Return value : none.
*/
void dl_fini()
{
        return;
}

/*
        Your own exec function : int dl_exec()
            char *result                : Reply value for 'result' key of JSON.
            struct agent_t *agent       : See above structure declaration.
            int ch_number               : Channel number where Index and Value match.
            json_t *json_obj            : It stores all indexes and values contained in the payload of
                                          the downstream message as Jansson C objects.
            unsigned char cloud_id      : Cloud Id.
            unsigned char sub_id        : Sub Cloud Id.
                                          These are the indications from which cloud they have
                                          been sent.
            char *rcv_header            : Contain the header information of the down stream
                                          message.
            char *rcv_payload           : Contain the paylod of the down stream message.
            char *hash                  : Hash value (MD5) of paylod of the down stream message.
          Return value : must be 0 for normal, -1 for error.
*/
int dl_exec(char *result, struct agent_t *agent, int ch_number, json_t *json_obj,
        unsigned char cloud_id, unsigned char sub_id,
        char *rcv_header, char *rcv_payload, char *hash)
{
        int rc;
        char *tx_payload;
        struct channel_t *ch;

        ch = (struct channel_t *)agent->ch[ch_number];

        pid_t pid;

//      The method to obtain Index and value from json_obj is shown below.
/*
        json_t *val;
        const char *key;
        val = json_object();
        json_object_foreach(json_obj, key, val) {
                if(json_is_string(val)) {
                        printf("%s", json_string_value(val));
                }
                else if(json_is_integer(val)) {
                        printf("%ld", json_integer_value(val));
                }
                else if(json_is_real(val)) {
                        printf("%lf", json_real_value(val));
```

```
                }
        }
*/

//      Write your own process here.

        snprintf(result, sizeof(char)*agent->buffer_size,
                "{YOUR OWN JSON OBJECT}");

        return(0);
}
```

4.5. Lua 言語スクリプト

Lua 言語スクリプトのサンプルコードを示します。

Lua 言語スクリプトは、luaL_dofile()によって PD Agent に取り込まれ、同スクリプトに含まれる関数の内、'lua_func' キーに指定された関数が実行されます。

指定された関数には、実行オブジェクトを実行する場合に環境変数として継承される全てのパラメータが継承(push)されます。 関数の戻り値 Stack1 は、正常終了であれば 0 異常終了であれば -1 を返して下さい。 戻り値の Stack2 は、応答メッセージの 'result'キーに与えられる JSON 文字列を返して下さい。

この、サンプルコードは、/usr/share/phhms/agent にインストールされています。

```
--
--   Copyright (c) 2018, 2019
--          Plat'Home CO., LTD. <support@plathome.co.jp>. All rights reserved.
--
-- Redistribution and use in source and binary forms, with or without
-- modification, are permitted provided that the following conditions
-- are met:
-- 1. Redistributions of source code must retain the above copyright
--      notice, this list of conditions and the following disclaimer.
-- 2. Redistributions in binary form must reproduce the above copyright
--      notice, this list of conditions and the following disclaimer in the
--      documentation and/or other materials provided with the distribution.
-- 3. Neither the name of the Plat'Home CO., LTD. nor the names of
--      its contributors may be used to endorse or promote products derived
--      from this software without specific prior written permission.
--
-- THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
-- IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
-- WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
-- ARE DISCLAIMED.   IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
-- INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
-- (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
-- SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
-- HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
-- LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
-- OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
-- SUCH DAMAGE.

-- Argument for lua function
--              string localname        : Local Device name (number).
```

```
--		string bind				: Pathname of UNIX domain socket to listen.
--		string push_to				: Pathname of UNIX domain socket to send.
--		number buffer_size	: Buffer size of UNIX domain socket.
--		number ch_number	: Channel number where Index and Value match.
--		boolean	reply				: Enable replay or not.
--		string name				: Name of chaanel.
--		string index				: Index of JSON string to be driven.
--		string value				: Value of JSON string to be driven.
--		number cloud_id				: Cloud Id.
--		number sub_id				: Sub Cloud Id.
--			These are the indications from which cloud they have
been sent.
--		string rcv_header	: Contain the header information of the down stream message.
--		string rcv_payload	: Contain the paylod of the down stream message.
--		string hash				: Hash value (MD5) of paylod of the down stream message.

-- Return value of lua function
--		Stack 1 : must be 0 for normal, -1 for error, will be get by lua_tointeger().
--		Stack 2 : any string for results, will be get by lua_tolstring().

function my_lua_func1(localname, bind, push_to, buffer_size, ch_number, reply,
		name, index, value, cloud_id, sub_id, rcv_header, rcv_payload, hash)

		fp = io.open("/tmp/my_lua_func1.txt", "w")

		s = string.format("localname : %s¥n", localname)
		fp:write(s)

		s = string.format("bind : %s¥n", bind)
		fp:write(s)

		s = string.format("push_to : %s¥n", push_to)
		fp:write(s)

		s = string.format("buffer_size : %d¥n", buffer_size)
		fp:write(s)

		s = string.format("ch_number : %d¥n", ch_number)
		fp:write(s)

		if (reply) then
			_replay = "true"
		else
			_replay = "false"
		end
		s = string.format("reply : %s¥n", _reply)
		fp:write(s)

		s = string.format("name : %s¥n", name)
		fp:write(s)

		s = string.format("index : %s¥n", index)
		fp:write(s)

		s = string.format("value : %s¥n", value)
		fp:write(s)

		s = string.format("cloud_id : %d¥n", cloud_id)
		fp:write(s)
```

```
        s = string.format("sub_id : %d¥n", sub_id)
        fp:write(s)

        s = string.format("rcv_header : %s¥n", rcv_header)
        fp:write(s)

        s = string.format("rcv_payload : %s¥n", rcv_payload)
        fp:write(s)

        s = string.format("hash : %s¥n", hash)
        fp:write(s)

        fp:close()

        -- reply message in "result" keys.
        result = string.format("{¥"done¥": true, ¥"localname¥":¥"%s¥"}", localname)

        return 0, result
end
```

4.6. 応答メッセージ

'reply' キーがtrue の場合、pd agent は'push to' に設定されるソケットに実行ステータスをJSON文字列で返します。

ステータス	応答メッセージ
実行オブジェクトを起動した	{“time”:”<timestamp>“,”repl_ to”:”<md5>“,”result”:”queuing”, “reason”:”matched”,”matched”:{“<key>“:”<value>“}}
実行オブジェクトの実行に失敗した	{“time”:”<timestamp>“,”repl_ to”:”<md5>“,”result”:”faild”, “reason”:”matched”,”matched”:{“<key>“:”<value>“}}
実行オブジェクトの実行を終えた	{“time”:”<timestamp>“,”repl_ to”:”<md5>“,”result”:”done”, “reason”:”matched”,”matched”:{“<key>“:”<value>“}}
一致するキーと値が存在しない	{“time”:”<timestamp>“,”repl_ to”:”<md5>“,”result”:”not queuing”, “reason”:”key and value notmatched”}
受信データが JSON フォーマットでない	{“time”:”<timestamp>“,”repl_ to”:”<md5>“,”result”:”not queuing”, “reason”:”not JSON form”}

PD Agentの応答メッセージ

ここで、<md5> はpd repeater から渡された受信データ(ペイロード) のMD5 ハッシュ値、'“<key>“:”<value>“は、一致した'index' キーと'value' キーに設定された条件です。
実行ステータスは実行オブジェクトを呼び出すexecv() の戻り値であり、実行オブジェクトの処理の結果を示すものではないため. システムに完全性を求めるのであれば実行オブジェクトで所定のUnixドメインソケットに応答を返すようにして下さい。
Dynamic Link モジュール使用時で、dl_exec() の文字列ポインタ引数 result が NULLで無い場合、実行終了時の’result’ キーの値は文字列ポインタ引数resultの値(JSON 文字列) に置き換えられます。

5. PD Handler Modbus

5.1. Modbus ファンクションコード

コード	名称	機能説明
0x01	Read Coils	デジタル出力に設定されているビット値を読み込む
0x02	Read Discrete Input	デジタル入力のビット値を読み込む.
0x03	Read Holding Registers	レジスタ出力に設定されている値を読み込む.
0x04	Read Input Registers	レジスタ入力の値を読み込む.
0x05	Write Single Coil	デジタル出力 1 bits のビット値を設定する.
0x06	Write Single Register	レジスタ出力 1 レジスタの値を設定する.
0x07	Read Exception Status	Modbus サーバ/クライアント間でエラーステータスを通知する.
0x09	Write Single Discrete Input	デジタル入力 1 bits のビット値を設定する.
0x0a	Write Single Input Register	レジスタ入力 1 レジスタの値を設定する.
0x0f	Write Multiple Coils	連続する複数のデジタル出力のビット値を設定する.
0x10	Write Multiple Registers	連続する複数のレジスタ出力の値を設定する.
0x11	Report Slave ID	接続可能なスレーブ(サーバ) 機器のユニット ID の一覧.
0x13	Write Multiple Discrete Input	連続する複数のデジタル入力のビット値を設定する.
0x14	Write Multiple Input Registers	連続する複数のレジスタ入力の値を設定する.
0x16	Mask Write Registers	レジスタ出力をマスクする.
0x17	Write And Read Registers	連続する複数のレジスタ出力の値を設定し、その値を読み込む.

PD Handler Modbus に用いられる Modbus ファンクションコード

0x09, 0x0a, 0x13, 0x14 は、物理的入力を持たないpd handler modbus server のデジタル入力もしくはレジスタ入力をクラウド側から設定できるよう用意された、本来のModbus プロトコルとは異なる独自仕様です。

Modbus ファンクションコードは、設定ファイルもしくはCSV ファイルあるいはクラウド側からの制御においてJSON 文字列の'function' キーの値に設定される. '0x' から始まる 16 進表記のファンクションコードを文字列として'function' キーの値に設定することも可能ですが、16 進表記とは別に整数表記と文字列表記を用いることも可能です。

コード	名称	整数表記	文字列表記
0x01	Read Coils	1	read_coils
0x02	Read Discrete Input	2	read_discrete_input
0x03	Read Holding Registers	3	read_holding_registers
0x04	Read Input Registers	4	read_input_registers
0x05	Write Single Coil	5	write_single_coil
0x06	Write Single Register	6	write_single_register
0x07	Read Exception Status	7	read_exception_status
0x09	Write Single Discrete Input	9	write_single_discrete_input
0x0a	Write Single Input Register	10	write_single_input_register
0x0f	Write Multiple Coils	15	write_multiple_coils
0x10	Write Multiple Registers	16	write_multiple_registers
0x11	Report Slave ID	17	report_slave_id
0x13	Write Multiple Discrete Input	19	write_multiple_discrete_input
0x14	Write Multiple Input Registers	20	write_multiple_iput_registers
0x16	Mask Write Registers	22	mark_write_registers
0x17	Write And Read Registers	23	write_and_read_registers

function キーに用いるファンクションコードの別称

5.2. PD Handler Modbus Client

5.2.1. デフォルトパス

PD Handler Modbus Client に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_handler_modbus_client	常駐実行オブジェクト（デーモン）
/lib/systemd/system/pd_handler_modbus_client.service	Systemd Service ファイル
/etc/init.d/pd_handler_modbus_client	RC ファイル
/var/webui/config/pd_handler_modbus_client.conf	設定ファイル
/var/webui/upload_dir/ pd_handler_modbus_client.csv	CSV ファイル
/var/run/pd_handler_modbus_client.pid	PID ファイル

PD Handler Modbus Client に関連するファイルのデォルトパス

5.2.2. 設定ファイルの書式

5.2.2.1. 構文

```
{
    “csv_file” : “<CSVファイルのパス名>“,
    “clients”: [
        {
            <clientsオブジェクト>,
            “acquisitions”:[
                {
                    < acquisitionsオブジェクト>
                },
                {
                    < acquisitionsオブジェクト>
                }
            ]
        },
        {
            <clientsオブジェクト>,
            “acquisitions”:[
                {
                    < acquisitionsオブジェクト>
                },
                {
                    < acquisitionsオブジェクト>
                }
            ]
        }
    ]
}
```

5.2.2.2. ルートオブジェクト

キー	データ型	説明
csv_file	文字列	CSVファイルのパス名. デフォルト値は '/var/webui/upload_dir/pd-data/pd_handler_modbus_client.csv'. (MAXPATHLEN)
clients	JSON obj	clients オブジェクト

ルートオブジェクト

5.2.2.3. clients オブジェクト

clients オブジェクトの配列数は、最大256 個です。Modbus のプロトコル(TCP, RTC) に依存するオブジェクトとプロトコルに依存しない共通のオブジェクトがあります。

キー	対象	データ型	説明
enable	共通	論理値	デフォルト値はfalse
localname		文字列	デバイスのローカル名(デバイス番号). (32byte)
memo		文字列	出力に付加されるユーザー定義文字列. (256byte)
bind		文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to		文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size		整数値	データのバッファサイズ(byte). デフォルト値は4096
receive		論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse
protocol		文字列	接続プロトコル'tcp' 又は'rtu' を指定. デフォルト値は'tcp'.
writeout		論理値	push to に指定するソケットに出力するか否か. デフォルト値は true.
interval		整数値	データを取得する間隔(sec). デフォルト値は 60.
timeout		整数値	データを取得出来ない場合のタイムアウト(msec). デフォルト値は 5000.
time_sysnc		論理値	時刻同期モード. デフォルト値は false
base_time		文字列	時刻同期モードの基準時刻を'HH:MM' 形式で指定する. デフォルト値は'00:00'
node	TCP	文字列	TCP 接続 PLC 機器の IP アドレス. デフォルト値は'127.0.0.1'
port		整数値	TCP 接続 PLC 機器のポート番号. デフォルト値は 502
device	RTU	文字列	シリアル接続PLC 機器のデバイス名. デフォルト値は'/dev/tty00'
rtu_speed		整数値	シリアル接続 PLC 機器のビットレート. デフォルト値は 115200
rtu_bits		整数値	シリアル接続 PLC 機器のビット数. 8 又は 7. デフォルト値は 8.
rtu_parity		文字列	シリアル接続PLC 機器のパリティ. 'none','even','odd' のいずれか. デフォルト値は'none'.
rtu_stop		整数値	シリアル接続PLC 機器のストップビット. 1 又は2. デフォルト値は1.
rtu_rts_delay		整数値	シリアル接続PLC 機器の RTS ディレイ値(usec). 0 の場合は自動設定される値.
rtu_reset		整数値	設定される値を越えて接続エラーが連続して発生した場合に 'device' キーに設定されるデバイスをリセットする. デフォルト値は 10.
memo	共通	文字列	出力に付加されるユーザー定義文字列. (256byte)
infos		JSON obj	出力に付加されるユーザー定義オブジェクト
acquisitions		JSON 配列	acquisitions オブジェクト

clients オブジェクト

5.2.2.4. acquisitions オブジェクト

キー	データ型	説明
unit	整数	PLC 機器のModbus ユニットID. 1 ～ 247 又は255 (TCP プロトコルのみ)
function	文字列又は整数	Modbus プロトコルのデータ読み出しファンクション名又はファンクション番号. 文字列の場合は、'read_coils', 'read_discrete input', 'read_holding_registers','read_input_registers' 又は'0x01'～'0x04'、整数値の場合は文字列の順に1～4. デフォルト値は 'read_holding_r_gisters'.
data_type	文字列又は整数	文字列の場合は'uint16_t','int16_t','uint32lsb_t','int32lsb_t', 'uint32msb_t','int32msb_t' のいずれか整数値の場合は文字列の順に0～5. function が'read_coils' 又は'read discrete_input'の場合は、'uint16_t' に固定される. デフォルト値は'uint16 t'.
address	文字列又は整数	読み込み介しレジスタアドレスを指定する. 文字列の先頭に'0x'が付加されている場合は16 進表記と解釈される. デフォルト値は'0x0'.
number	文字列又は整数	読み込むビット数またはレジスタ数、文字列の先頭に'0x' が付加されている場合は16 進表記と解釈される. data_type が'uint32lsb_t', 'int32ls_ t','uint32msb_t','int32msb_t' の場合は、内部的に2 倍の値で扱われる. デフォルト値は1.

acquisitions オブジェクト

5.2.3. 利用可能なファンクションコード

PD Handler Modbus Client で利用可能なファンクションコードを示します。

コード	名称	ローカル	クラウド
0x01	Read Coils	○	○
0x02	Read Discrete Input	○	○
0x03	Read Holding Registers	○	○
0x04	Read Input Registers	○	○
0x05	Write Single Coil		○
0x06	Write Single Register		○
0x07	Read Exception Status		
0x09	Write Single Discrete Input		
0x0a	Write Single Input Register		
0x0f	Write Multiple Coils		○
0x10	Write Multiple Registers		○
0x11	Report Slave ID		○
0x13	Write Multiple Discrete Input		
0x14	Write Multiple Input Registers		
0x16	Mask Write Registers		
0x17	Write And Read Registers		○

PD Handler Modbus Client で利用可能なファンクションコード

ローカルは、設定ファイルもしくはCSV ファイルに指定可能なファンクションコード、クラウドはクラウド側か指定可能なファンクションコードを意図します。

5.2.4. CSV ファイル

/var/webui/config/pd_handler_modbus_client.conf の csv_file キーに設定される CSV ファイルを置くことで、/var/webui/config/pd_handler_modbus_client.conf で設定された１つのデバイス番号に対し複数の取得 Modbus クライアントデバイスを設定することができます。
CSV ファイルの書式は、次の通りです。

デバイス番号，ユニット ID，読込方式，データタイプ，読込開始アドレス，読込レジスタ数

パラメータ	データの型式	説明
デバイス番号	半角英数字	clients オブジェクトの localname キーに設定されたデバイス番号を記載します。　設定されていないデバイス番号は無視されます。 先頭が'#'または"/"の場合は、コメント行として扱われます。
ユニット ID	半角数字	PLC 機器の Modbus ユニット ID を設定します。ユニット ID は、1～247 または 255 を記載します。
読込方式	半角英数字	読込方式として次の何れかを記載します。 デジタル出力：'read_coils'　又は '0x01' 又は '1' デジタル入力：'read_discrete_input' 又は '0x02' 又は '2' レジスタ出力：'read_holding_registers' 又は '0x03' 又は '3' レジスタ入力：'read_input_registers' 又は'0x04' 又は '4'
データタイプ	半角英数字	データタイプとして次の何れかを記載します。 符号なし 16 ビット整数：　'uint16_t' 又は '0' 符号付き 16 ビット整数：　'int16_t' 又は '1' 符号なし 32 ビット整数/リトルエンディアン：'uint32lsb_t' 又は '2' 符号付き 32 ビット整数/リトルエンディアン：'int32lsb_t' 又は '3' 符号なし 32 ビット整数/ビッグエンディアン：'uint32msb_t' 又は '4' 符号付き 32 ビット整数/ビッグエンディアン：'int32msb_t' 又は '5'
読込開始アドレス	半角英数字	読み込みたいデータが格納されている PLC 機器上の開始アドレスを設定します。　先頭が'0x'の場合は 16 進数と解釈されます。
読込レジスタ数	半角数字	読み込みたいレジスタ数を記載します。

PD Handler Modbus Client の CSV ファイルの書式

各パラメータの区切りはカンマ、先頭がシャープ’#’もしくはスラッシュ’/’の行はコメント行と見なされます。

CSV ファイルが読み込まれると clients オブジェクトに設定された Modbus クライアントデバイスは上書きされます。そのため、CSV ファイルには clients オブジェクトに設定した Modbus クライアントデバイスを含む取得したい全ての Modbus クライアントデバイスを記載して下さい。

5.2.5. 基準時刻制御

基準時刻制御は、特定の時刻にデータを取得する機能です。
clients オブジェクトの time_sysnc キーを true に設定し、interval キーと base_time キーで取得間隔と取得時刻を設定します。
基準時刻制御における「取得時間間隔」は 300, 600, 900, 1800, 3600, 7200, 10800, 14400, 21600, 28800, 43200 と 86400 の倍数に限られます。「取得時間間隔」としてこれら以外の値が設定されると PD Handler Modbus Client 内で次のように扱われます。

取得時間間隔の設定値	実動作値
0 ～ 599	300
600 ～ 899	600
900 ～ 1799	900
1800 ～ 3599	1800
3600 ～ 7199	3600
7200 ～ 10799	7200
10800 ～ 14399	10800
14400 ～ 21599	14400
21600 ～ 28799	21600
28800 ～ 43199	28800
43200 ～ 86399	43200
86400～	86400 の倍数

基準時刻制御における取得時間間隔の設定と実動作値

「基準時刻」とは動作の起点となる時刻で、例えば「取得時間間隔」を 300 とし、「基準時刻」を”00:01”とした場合、データの取得は 00:01, 00:06, 00:11 ... 00:56, 01:01 ... 23:56, 00:01 の定刻に行われます。
データの取得開始時刻は「基準時刻」に設定さた時刻そのものではなく、「基準時刻」と「取得時間間隔」から算定される直近の時刻となります。
例えば 08:30 に「基準時刻」”01:05”、 「取得時間間隔」10800 の設定が行われた場合、最初のデータ取得は 10:05 に行われ、以降 13:05, 16:05, 19:05, 22:05, 01:05 の順におこなわれます。

5.2.6. メッセージオブジェクト

入出力メッセージに用いられる JSON オブジェクトを示します。

キー	内容	データ型	方向	備考
time	データ取得日時	文字列	出力	ISO8601 拡張書式
prototocol	プロトコル	文字列	入出力	'tcp' 又は 'rtu'
node	取得先 IP アドレス	文字列	入出力	'tcp' 時のみ
port	ポート番号	整数値	入出力	'tcp' 時のみ
device	デバイスファイル名	文字列	入出力	rtu' 時のみ
unit	Modbus Unit ID	整数値	入出力	
memo	メモ	文字列	出力	memo キーに設定された値
address	読み／書き込みアドレス	整数値	入出力	
number	読み込むレジスタ数	整数値	入力	
function	Modbus function code	整数値	入出力	
data_type	データの型	文字列	入出力	'uint16_t', 'int16_t', 'uint32lsb_t', 'int32lsb_t', 'uint32msb_t', 'int32msb_t' のいずれか
valuse	読み／書き込み値	整数配列	入出力	
reply_to	リクエストメッセージの MD5	文字列	出力	
result	制御ステータス	文字列	出力	'done', 'not queuing', 'failed' のいずれか
reason	エラーの理由	文字列	出力	

PD Handler Modbus Client のメッセージオブジェクト

5.2.7. クラウドからの制御

クラウドから PD Repeater を介し制御(JSON)文字列を送ることで、接続されている PLC 機器のレジスタを読み書きすることができます。
制御文字列には、PLC 機器を特定するため clients オブジェクトの protocol, node, port, もしくは device と acquisitions オブジェクトの unit, function, data_type, address, number および書き込みファンクションにおいては、valuse に値を記載します。
node, port, もしくは device は、設定ファイルに設定されているものでなければなりません。物理的に接続されていても設定ファイルに設定されていない node, port, もしくは device を操作することはできません。

例えば、TCP 接続されている PLC 機器の入力レジスタを読み込むのであれば、その制御文字列は次のようになります。

```
{
    "protocol":"tcp", "node":"192.168.1.8", "port":"502",
     "unit":255,"function":"0x04",
      "data_type":"uint16_t","address":"0x160","number":5
}
```

これに対する、応答メッセージは、次のようになります。


```
{
    "time":"2017-09-05T15:30:05.758+09:00",
    "reply_to":"452556d8daf1f7eb483d00ee03718e8e","result":"done",
    "memo":"Modbus Client 00",
    "protocol":"tcp","node":"192.168.1.8","port":502,
    "unit":255,"address":352,"function":4,"data_type":"uint16_t",
    "values":[65535,0,1,2,3]
}
```


ここで、reply_to は、制御文字列の MD5 です。

シリアル接続の PLC 機器の出力レジスタを書き込むのであれば、その制御文字列は次のようになります。


```
{
    "protocol":"rtu","device":"/dev/ttyMDF1",
    "unit":16,"function":"0x10","data_type":"uint32lsb_t","address":"0x120",
    "values":[4567,891011]
}
```


ここで data_type が 32bit の場合、values は上位/下位の 16 bits に分割されて処理されるため、書き込む値の数が 1 であっても function は Write Multiple Registers を使用しなくてはなりません。

これに対する、応答メッセージは、次のようになります。


```
{
    "time":"2017-09-05T16:35:13.653+09:00",
    "reply_to":"fe9b49ff045f435a371303a82281f3f9","result":"done",
    "memo":"Modbus Client 01",
    "protocol":"rtu","device":"/dev/ttyMDF1",
    "unit":16,"address":288,"function":16,"data_type":"uint32_t",
    "values":[4567,891011]
}
```

5.3. PD Handler Modbus Server

Modbus サーバーは次表に示すレジスタマップを保持し、PLC 機器からの Modbus プロトコルによる接続を待ち受け、PLC 機器の書き込み操作によりレジスタの値を更新すると共に更新されたレジスタとその値を PD Repeater を介してクラウドに送ります。
また、PD Repeater を介してクラウドから送られる制御メッセージ(JSON 文字列)に基づきレジスタマップを読み書きすることもできます。
レジスタマップは、60 秒毎に更新があればレジスタマップファイル registers.map に出力されます。

レジスタ	開始アドレス	サイズ
デジタル出力(Coils)	0x000	uint8 _t × 2048
デジタル入力(Discrete Input)	0x000	uint8_t × 2048
レジスタ出力(Holdig Registers)	0x000	uint16_t × 2048
レジスタ入力(Input Registers)	0x000	uint16_t × 2048

PD Handler Modbus Server のレジスタマップ

5.3.1. デフォルトパス

PD Handler Modbus Server に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_handler_modbus_server	常駐実行オブジェクト（デーモン）
/usr/sbin/pd_handler_modbus_register	レジスタマップ作成／ダンプツール
/lib/systemd/system/pd_handler_modbus_server.service	Systemd Service ファイル
/etc/init.d/pd_handler_modbus_server	RC ファイル
/var/webui/config/pd_handler_modbus_server.conf	設定ファイル
/var/webui/.modbus_server/registers.map	レジスタマップファイル
/var/run/pd_handler_modbus_server.pid	PID ファイル

PD Handler Modbus Server に関連するファイルのデォルトパス

5.3.2. 設定ファイルの書式

5.3.2.1. 構文

```
{
    “registers_file” : “<レジスタマップファイルのパス名>“,
    “servers”: [
        {
            <serversオブジェクト>
        },
            <serversオブジェクト>
        }
    ]
}
```

5.3.2.2. ルートオブジェクト

キー	データ型	説明
registers_file	文字列	レジスタマップファイルのパス名. デフォルト値は '/var/webui/.modbus_server/pd-data/registers.map'. (MAXPATHLEN)
servers	JSON obj	servers オブジェクト

ルートオブジェクト

5.3.2.3. servers オブジェクト

server オブジェクトは、PD Handler Modbus Serverの動作を規定する設定オブジェクトです。 server オブジェクトの配列数は、最大8 個. Modbus のプロトコル(TCP, RTC) に依存するオブジェクトとプロトコルに依存しない共通のオブジェクトがあります。

キー	対象	データ型	説明
enable	共通	論理値	デフォルト値はfalse
localname		文字列	デバイスのローカル名(デバイス番号). (32byte)
memo		文字列	出力に付加されるユーザー定義文字列. (256byte)
bind		文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/＜デバイス番号＞.sock が設定されます.
push_to		文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
buffer_size		整数値	データのバッファサイズ(byte). デフォルト値は4096
receive		論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse
protocol		文字列	接続プロトコル'tcp' 又は'rtu' を指定. デフォルト値は'tcp'.
writeout		論理値	push to に指定するソケットに出力するか否か. デフォルト値は true.
timeout		整数値	データを取得出来ない場合のタイムアウト(msec). デフォルト値は 5000.
node	TCP	文字列	TCP 接続待ち受け IP アドレス. デフォルト値は'127.0.0.1'
port		整数値	TCP 接続待ち受けポート番号. デフォルト値は 502
device	RTU	文字列	シリアル接続のデバイス名. デフォルト値は'/dev/tty00'
rtu_speed		整数値	シリアル接続のビットレート. デフォルト値は 115200
rtu_bits		整数値	シリアル接続のビット数. 8 又は 7. デフォルト値は 8.
rtu_parity		文字列	シリアル接続のパリティ. 'none','even','odd' のいずれか. デフォルト値は'none'.
rtu_stop		整数値	シリアル接続のストップビット. 1 又は2. デフォルト値は1.
rtu_rts_delay		整数値	シリアル接続のRTS ディレイ値(usec). 0 の場合は自動設定される値.
unit		整数値	自身に付与するModbus ユニットID. 整数値1 ～ 247.
memo	共通	文字列	出力に付加されるユーザー定義文字列. (256byte)
infos		JSON obj	出力に付加されるユーザー定義オブジェクト

servers オブジェクト

5.3.3. 利用可能なファンクションコード

PD Handler Modbus Server で利用可能なファンクションコードを示します。

コード	名称	ローカル	クラウド
0x01	Read Coils	○	○
0x02	Read Discrete Input	○	○
0x03	Read Holding Registers	○	○
0x04	Read Input Registers	○	○
0x05	Write Single Coil	○	○
0x06	Write Single Register	○	○
0x07	Read Exception Status	○	
0x09	Write Single Discrete Input		○
0x0a	Write Single Input Register		○
0x0f	Write Multiple Coils	○	○
0x10	Write Multiple Registers	○	○
0x11	Report Slave ID	○	○
0x13	Write Multiple Discrete Input		○
0x14	Write Multiple Input Registers		○
0x16	Mask Write Registers	○	
0x17	Write And Read Registers	○	○

PD Handler Modbus Server で利用可能なファンクションコード

ローカルは、ローカルに接続するPLC 機器からに指定可能なファンクションコード、クラウドはクラウド側か指定可能なファンクションコードを意図します。

5.3.4. クラウドからからの制御

クラウドから PD Repeater を介し制御(JSON)文字列を送ることで、レジスタマップを読み書きすることができます。
acquisitions オブジェクトの function, data_type, address, number 相当のパラメタと、書き込みファンクションにおいては、valuse に値を記載します。
PD Handler Modbus Client とは異なり自身のレジスタマップを読み書きするため、node, port もしくは device の設定は必要としません。

例えば、入力レジスタを読み込むのであれば、その制御文字列は次のようになります。

```
{
    "function":"0x04","data_type":"uint16_t","address":"0x160","number":5
}
```

これに対する応答メッセージは、次のようになります。


```
{
    "time":"2017-09-05T15:30:05.758+09:00",
    "reply_to":"14239b236795a8a9e06d60a25c015f2e","result":"done",
    "memo":"Modbus Server 00",
    "protocol":"tcp","node":"192.168.1.8","port":502,
    "unit":255,"address":352,"function":4,"data_type":"uint16_t",
    "values":[65535,0,1,2,3]
}
```


ここで、reply_to は、制御文字列の MD5 です。
レジスタマップに node, port, もしくは device の区別はありませんが、PD Repeater からは異なる localname を持つ個別のデバイスと位置付けられるため、応答メッセージには、制御文字列を受けだ Unix ドメインソケットと対をなす node, port, もしくは device が付加されます。

入力レジスタを書き込むのであれば、その制御文字列は次のようになります。


```
{
    "function":"0x14","data_type":"uint32_t","address":"0x140",
    "values":[4567,8910,561,435]
}
```


これに対する応答メッセージは、次のようになります。


```
{
    "time":"2017-09-05T16:35:13.653+09:00",
    "reply_to":"d5bcc347ad36fa6e7090d0f763108882","result":"done",
    "memo":"Modbus Server 01",
    "protocol":"rtu","device":"/dev/ttyMDF1",
    "unit":17,"address":320,"function":20,"data_type":"uint32_t",
    "values":[4567,8910,561,435]
}
```

5.3.5. レジスタマップのダンプと作成

レジスタマップファイル(/var/webui/.modbus_server/registers.map)の内容は、/usr/sbin/pd_handler_modbus_register コマンドを用いて CSV ファイルへダンプもしくは、CSV ファイルからレジスタマップファイルを作成することができます。
pd_handler_modbus_register コマンドのコマンドオプションを示します。

```
root# /usr/sbin/pd_handler_modbus_register -h
usages: pd_handler_modbus_register [-d][-f <mapfile>][-C <CSV file>]
 -d                Dump a registers map file to CSV file.
 -f mapfile        Registers map file. [default: /var/webui/.modbus_server/registers.map]
 -C CSV file       CSV file path for in/out [default: stdin/stdout]
```

レジスタマップファイルの CSV ファイルへダンプ例を示します。

```
root# /usr/sbin/pd_handler_modbus_register -d
#address, coil, discrete_input, register, input_register, comment
0,0,0,0,0,
1,1,0,0,0,
2,0,0,65534,0,
3,0,0,0,0,
4,1,0,3,0,
5,0,1,2,0,
6,1,0,1,0,
7,0,1,0,65534,
8,0,0,65535,65535,
9,0,1,65534,0,
10,0,0,0,1,
11,0,1,0,0,
12,0,0,0,0,
13,0,0,0,0,
14,0,0,0,0,
15,0,0,0,0,
      :
2047,,0,0,0,0,
```

ここで、数値の並びは、

アドレス(0～2047)、ディジタル出力値、ディジタル入力値、レジスタ出力値、レジスタ入力値
の順です。

値は全て符号なし 16 ビットです。
レジスタマップファイルはコメントを保持しないため、コメント欄は常に空欄となります。

CSV ファイルからレジスタマップファイルを作成する場合は、次のような CVS ファイルを用意し、pd_handler_modbus_register コマンドでレジスタマップファイルに変換します。

```
// default register map
#address, coil, discrete_input, register, input_register, comment
 0,0,0,0,0,
 1,1,1,1,1,
 8,0,1,65535,65534,
 16,0,0,32767,65535,
 17,0,0,65535,65535,
```

CSV ファイルは、アドレス 0～2047 の全てを用意する必要はなく、値を設定する必要があるアドレスだけで構いません。
値は全て符号なし 16 ビットで指定し、ディジタル出力値とディジタル入力値は必ず 0 又は 1 とします。
記載されていないアドレスのレジスタ値は 0 がセットされます。
先頭が '¥#' もしくは '/' の行はコメントと見なされます。

pd_handler_modbus_register コマンドによるレジスタマップファイルへの変換方法は次の通りです。

```
root# /usr/sbin/pd_handler_modbus_register -C default_register_map.csv
```

6. PD Handler BLE (Node.js)

6.1. デフォルトパス

PD Handler BLE (Node.js)に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/opt/pd/handler/pd-handler.js	実行ファイル
/etc/init.d/pd-handler-ble-js	RC ファイル
/var/webui/config/pd-handler-ble.conf	設定ファイル
/var/run/pd-handler-ble-js.pid	PID ファイル
/var/webui/pd-logs/pd-handler-ble-js.log	ログファイル
/var/werbui/.blebackup/pd-handler-ble.conf	設定ファイルのバックアップ
/var/werbui/.blebackup/pd-handler-ble-js-restore.log	リストア用ファイル
/var/webui/.bsensor/xxx.json	local 用センサデータファイル(xxx is localname.)

PD Handler BLE (Node.js)に関連するファイルのデォルトパス

6.2. 設定ファイルの書式

6.2.1. 構文

```
{
    “servers”: <servers オブジェクト>,
    “beacon”: <beacon オブジェクト>,
    “blesensor”: <blesensor オブジェクト>
}
```

6.2.2. ルートオブジェクト

キー	データ型	説明
servers	JSON obj	serversオブジェクト
beacon	JSON obj	beacon オブジェクト
blesensor	JSON 配列	blesensor オブジェクト

ルートオブジェクト

6.2.3. servers オブジェクト

キー	データ型	説明
json_ldir	文字列	データ用ディレクトリパス
handler_ldir	文字列	バックアップ用ディレクトリパス
blehandler_monitor_api	文字列	beacon モニタリング用 API ファイル

serevers オブジェクト

6.2.4. beacon オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
duplicate_type	文字列	“interval”, “entry”, “inout”
duplicate_interval	整数値	重複制御時間間隔. 0 ~ 3600000[msec]
appendix_info	文字列	付随情報.デフォルト値はシリアルナンバー.
payload_manage	JSON obj	ペイロード管理
data_filter_rule	JSON 配列	データフィルタ
rssi_filter	整数値	RSSI フィルタ
add_data	JSON obj	ユーザー定義情報
enable	論理値	データを送信するか否か

beacon オブジェクト

6.2.4.1. payload_manage オブジェクト

キー	データ型	説明
data	論理値	送信データにアドバタイズデータを付与するか否か
localname	論理値	送信データにローカルネームを付与するか否か
type	論理値	送信データにビーコンタイプ(iBeacon)を付与するか否か

payload_manage オブジェクト

6.2.4.2. data_filter_rule オブジェクト

キー	データ型	説明
length	整数値	prefixの長さ
prefix	文字列	前方一致のデータフィルタ

data_filter_rule オブジェクト

6.2.5. blesensor オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
uuid	文字列	Bluetooth Device Address(コロンを削除し、16 進数を英小文字で表した文字列)
memo	文字列	出力に付加されるユーザー定義文字列. (256byte)
interval	整数値	データ送信間隔.[msec]
txpower	整数値	送信出力[dBm]
local	論理値	送信先設定の本体内(local)へ送信するか否か
enable	論理値	データを送信するか否か

blesensor オブジェクト

7. PD Handler BLE (C)

7.1. デフォルトパス

PD Handler BLE (C)に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd-handler-ble-c	常駐実行オブジェクト（デーモン）
/etc/init.d/pd-handler-ble-c	RC ファイル
/var/webui/config/pd-handler-ble.conf	設定ファイル
/var/run/pd-handler-ble-c.pid	PID ファイル
/var/webui/pd-logs/pd-handler-ble-c.log	ログファイル
/var/werbui/.blebackup/pd-handler-ble.conf	設定ファイルのバックアップ
/var/werbui/.blebackup/pd-handler-ble-c-restore.log	リストア用ファイル
/var/webui/.bsensor/xxx.json	local 用センサデータファイル(xxx is localname.)
/opt/pd/lua/ble/devices/*.lua	Lua 拡張用ファイル

PD Handler BLE (C)に関連するファイルのデォルトパス

7.2. 設定ファイルの書式

設定ファイルは PD Handler BLE (Node.js)と同じです。

詳細は 6.2.設定ファイルの書式を参照してください。

8. PD Handler UART

8.1. デフォルトパス

PD Handler UART に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd-handler-uart	常駐実行オブジェクト（デーモン）
/etc/init.d/pd-handler-uart	RC ファイル
/var/webui/config/pd-handler-uart.conf	設定ファイル
/var/run/pd-handler-uart.pid	PID ファイル
/var/webui/pd-logs/pd-handler-uart.log	ログファイル
/opt/pd/lua/uart/enocean/devices/*.lua	Lua 拡張用ファイル

PD HandlerUART に関連するファイルのデォルトパス

8.2. 設定ファイルの書式

8.2.1. EnOcean

8.2.1.1. 構文

```
{
    “prototype”:”enocean”,
    “enocean”: <enocean オブジェクト> ,
    “enocean_device”: <enocean_device オブジェクト>
}
```

8.2.1.1. ルートオブジェクト

キー	データ型	説明
prototype	文字列	“enocean”
enocean	JSON obj	enocean オブジェクト
enocean_device	JSON 配列	enocean_device オブジェクト

ルートオブジェクト

8.2.1.2. enocean オブジェクト

キー	データ型	説明
serial_port	文字列	デフォルトは”/dev/ttyEX2”
raw_data_mode	論理値	true なら受信データをそのまま送信

enocean オブジェクト

8.2.1.3. enocean_device オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
deviceid	文字列	ID
memo	文字列	出力に付加されるユーザー定義文字列. (256byte)
eep	文字列	EnOcean Equipment Profiles
enable	論理値	データを送信するか否か

enocean_device オブジェクト

9. PD Handler HVSMC

9.1. デフォルトパス

PD Handler HVSMC に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd-handler-hvsmc	常駐実行オブジェクト（デーモン）
/etc/init.d/pd-handler-hvsmc	RC ファイル
/var/webui/config/pd-handler-hvsmc.conf	設定ファイル
/var/run/pd-handler-hvsmc.pid	PID ファイル
/var/webui/pd-logs/pd-handler-hvsmc.log	ログファイル

PD Handler HVSMC に関連するファイルのデォルトパス

9.2. 設定ファイルの書式

9.2.1. 構文

```
{
        “hvsmc”: <hvsmc オブジェクト>
}
```

9.2.2. ルートオブジェクト

キー	データ型	説明
hvsmv	JSON obj	hvsmc オブジェクト

ルートオブジェクト

9.2.3. hvsmc オブジェクト

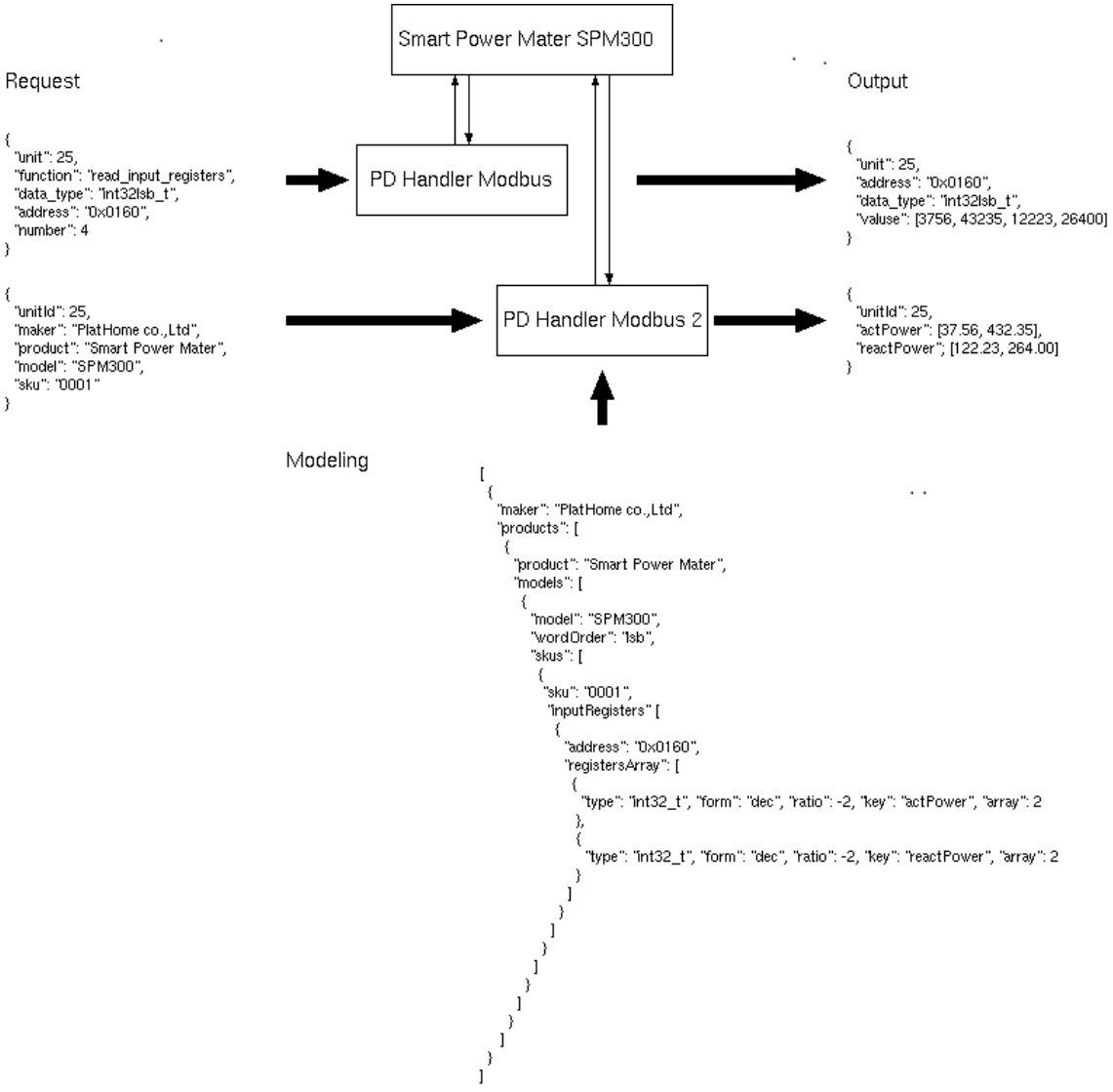
キー	データ型	説明
enable	論理値	データを送信するか否か. デフォルト値は false
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
memo	文字列	出力に付加されるユーザー定義文字列. (256byte)
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
nif	文字列	高圧スマート電力量メータを接続する Ethernet インタフェースのデバイス名.
raw_data_mode	論理値	取得したデータを変換せずに PD Repeater へ送るか否か. デフォルト値は false
demand_info	論理値	定時計測時に需要電量の取得を行うか否か. デフォルト値は false
measured_info	論理値	計測値の取得を行うか否か. デフォルト値は false
measured_interval	整数値	計測値の取得を行う時間間隔（min）. デフォルト値は 0

hvsmc オブジェクト

10. PD Handler Modbus 2

10.1. Modbus モデリングファイル

従来の PD Handler Modbus では、Modbus のファンクションコードと共にデータのタイプとレジスタのアドレスと個数を直接指定することで、データを整数配列として読み書きしていたのに対し、PD Handler Modbus 2 では、モデリングファイルを用意することで、そのインデックスとなるメーカー／プロダクト／モデル／SKU を指定することにより、モデルリングファイルの定義に従いデータの読み書きを行います。



PD Handler Modbus と PD Handler Modbus 2 の比較

PD Handler Modbus 2 を使用するためには、モデリングファイルを用意しなくてはなりませんが、複数のレジスタセットや混在するデータタイプに対するアクセスを一つのインデックスを読み出すことで処理でき、また、JSON キーを自由に割り当てることができ、データの書式も幾つか選択できるようになりました。

10.1.1. インデックスの階層

モデリングファイルのインデックスは、以下の JSON 文字列に示すようにメーカー名、プロダクト名、モデル名、SKU 名の 4 階層で構成され、各階層が JSON オブジェクトの配列となっており、各配列要素を maker オブジェクト、product オブジェクト、model オブジェクト、sku オブジェクトと呼びます。sku オブジェクトには、レジスタと出力書式を定義する幾つかのオブジェクト配列が含まれています。

```
[
    {
        "maker": "Maker Name A",
        "products": [
            {
                "product": "Product Name",
                "models":[
                    {
                        "model": "Model Name",
                        "skus": [
                            {
                                "sku": "Sku Name",
                                    :
                            }
                        ]
                    }
                ]
            }
        ]
    },
    {
        "maker": "Maker Name B",
                :
    }
]
```

なお、PD Handler Modbus 2 には、複数のモデリングファイルをマージするツールが用意されているため、モデリングファイルはSKUやモデル毎に分けて作成することも可能です。

10.1.2. モデリングファイルの例

モデリングファイルの例を示します。

この例では、1 つのファイルに 2 つの SKU が記載されています。

SKU '0001' は、電流と電圧を配列（ex. "current":[23.1, 45.2], "voltage":[100.2, 100.3]）で出力するのに対し、SKU '0002' は個別（ex.　"current0":23.1, "current1":45.2, "voltage0":100.2, "voltage1":100.3）に出力します。

```
{
    "maker":"Example Co.,Ltd.",
    "include":false,
    "products":[
        {
            "product":"Smart Meter",
            "include":false,
            "models":[
                {
                    "model":"ZZZ1",
                    "include":false,
                    "byteOrder":"lsb",
                    "wordOrder":"lsb",
                    "bitOrder":"msb",
                    "precision":15,
                    "skus":[
                        {
                            "sku":"0001",
                            "include":false,
                            "outputBits":[
                                {
                                    "address":"0x0101",
                                    "enable":false,
                                    "writeOnly":true,
                                    "bitsArray":[
                                        { "include":true, "key":"switch" }
                                    ]
                                }
                            ],
                            "inputRegisters":[
                                {
                                    "address":"0x0001",
                                    "enable":true,
                                    "registersArray":[
                                        { "include":true, "type":"int32_t",
                                          "form":"dec", "ratio":0, "key":"power" },
                                        { "include":false, "type":"int32_t",
                                          "form":"dec", "ratio":0, "key":"reversePower" },
                                        { "include":true, "type":"int32_t",
                                          "form":"dec", "ratio":-1,"key":"current", "array":2 },
                                        { "include":true, "type":"int32_t",
                                          "form":"dec", "ratio":-1,"key":"voltage", "array":2 },
                                        { "include":false, "type":"int32_t",
                                          "form":"dec", "ratio":0, "key":"instPower" },
                                        { "include":false, "type":"uint32_t",
                                          "form":"dec", "ratio":0, "key":"serialNumber"}
                                    ]
                                }
```

```
                ],
                "infos": {}
            },
            {
                "sku":"0002",
                "include":false,
                "outputBits":[
                    {
                        "address":"0x0101",
                        "enable":false,
                        "writeOnly":true,
                        "bitsArray":[
                            { "include":true, "key":"switch" }
                        ]
                    }
                ],
                "inputRegisters":[
                    {
                        "address":"0x0001",
                        "enable":true,
                        "registersArray":[
                            { "include":true, "type":"int32_t",
                              "form":"dec", "ratio":0, "key":"power" },
                            { "include":false, "type":"int32_t",
                              "form":"dec", "ratio":0, "key":"reversePower" },
                            { "include":true, "type":"int32_t",
                              "form":"dec", "ratio":-1,"key":"current0" },
                            { "include":true, "type":"int32_t",
                              "form":"dec", "ratio":-1,"key":"current1" },
                            { "include":true, "type":"int32_t",
                              "form":"dec", "ratio":-1,"key":"voltage0" },
                            { "include":true, "type":"int32_t",
                              "form":"dec", "ratio":-1,"key":"voltage1" },
                            { "include":false, "type":"int32_t",
                              "form":"dec", "ratio":0, "key":"instPower" },
                            { "include":false, "type":"uint32_t",
                              "form":"dec", "ratio":0, "key":"serialNumber" }
                        ]
                    }
                ],
                "infos": {}
            }
        ]
    }
]
}
```

PD Handler Modbus 2 が読み込むモデリングファイルのルートオブジェクトは、maker オブジェクトの配列でなければなりませんが、モデリングファイルをマージするツールは以上の例のようにルートオブジェクトに maker オブジェクトを配置したファイルも処理することができます。

10.1.3. Modbus モデリングファイルの書式

10.1.3.1. 構文

PD Handler Modbus 2 のモデリングファイルは次の書式で構成されています。

```
[
    {
        maker オブジェクト,
        "products": [
            {
                product オブジェクト,
                models": [
                    {
                        model オブジェクト,
                        skus": [
                            {
                                sku オブジェクト,
                                "outputBits": [
                                    {
                                        bit オブジェクト,
                                        "bitsArray":[
                                            {
                                                bitsArray オブジェクト,
                                            },
                                            {
                                                bitsArray オブジェクト,
                                            }
                                        ]
                                    },
                                    {
                                        bit オブジェクト,
                                            :
                                    }
                                ],
                                "inputBits": [
                                    {
                                        bit オブジェクト,
                                            :
                                    },
                                    {
                                        bit オブジェクト,
                                            :
                                    }
                                ],
                                "outputRegisters": [
                                    {
                                        register オブジェクト,
                                        "registersArray":[
                                            {
                                                registersArray オブジェクト,
                                            },
                                            {
                                                registersArray オブジェクト,
                                            }
```

```
                                        ]
                                    },
                                    {
                                        register オブジェクト,
                                            :
                                    }
                                ],
                                "inputRegisters": [
                                    {
                                        register オブジェクト,
                                            :
                                    },
                                    {
                                        register オブジェクト,
                                            :
                                    }
                                ],
                                "rawRequests": [
                                      rawRequest オブジェクト,
                                            :
                                ],
                                "infos": infos オブジェクト,
                                "infos_poll": infos_poll オブジェクト,
                                "infos_od": infos_od オブジェクト
                            },
                            {
                                sku オブジェクト,
                                    :
                            }
                        ]
                    },
                    {
                        model オブジェクト,
                            :
                    }
                ]
            },
            {
                product オブジェクト,
                        :
            }
        ]
    },
    {
        maker オブジェクト,
            :
    }
]
```

10.1.3.2. ルートオブジェクト

キー	データ型	説明
	JSON 配列	maker オブジェクトの配列. (最大 32 オブジェクト)

ルートオブジェクト

10.1.3.3. maker オブジェクト

キー	データ型	説明
maker	文字列	メーカー名. (64byte)
include	論理値	メーカー名を送信するデータに含むか否か. デフォルト値は true.
products	JSON配列	product オブジェクトの配列. (最大 32 オブジェクト)

maker オブジェクト

10.1.3.4. product オブジェクト

キー	データ型	説明
product	文字列	プロダクト名. (64byte)
include	論理値	プロダクト名を送信するデータに含むか否か. デフォルト値は true.
models	JSON配列	model オブジェクトの配列. (最大 32 オブジェクト)

product オブジェクト

10.1.3.5. model オブジェクト

キー	データ型	説明
model	文字列	モデル名. (64byte)
include	論理値	モデル名を送信するデータに含むか否か. デフォルト値は true.
wordOrder	文字列	ワードの順列. 'lsb' 又は 'msb'. デフォルト値は 'lsb'.
byteOrder	文字列	バイトの順列. 'lsb' 又は 'msb'. デフォルト値は 'lsb'.
bitOrder	文字列	ビットの順列. 'lsb' 又は 'msb'. デフォルト値は 'lsb'.
wait	整数値	シリアル接続 PLC 機器に対する読み書きコマンド後の待ち時間 (μ sec). デフォルト値は 0.
precision	整数値	送信するデータとしてJSON文字列にダンプされる際のリアル型データの精度(libjanssonに与えるパラメータ) 1～17. デフォルト値は 17.
skus	JSON配列	sku オブジェクトの配列. (最大 32 オブジェクト)

model オブジェクト

10.1.3.6. sku オブジェクト

キー	データ型	説明
model	文字列	sku 名. (64byte)
include	論理値	sku名を送信するデータに含むか否か. デフォルト値は true.
outputBits	JSON配列	出力ビットのレジスタ情報. bit オブジェクトの配列 (最大 32 オブジェクト)
inputBits	JSON配列	入力ビットのレジスタ情報. bit オブジェクトの配列 (最大 32 オブジェクト)
outputRegisters	JSON配列	出力レジスタのレジスタ情報. register オブジェクトの配列 (最大 32 オブジェクト)
inputRegisters	JSON配列	入力レジスタのレジスタ情報. register オブジェクトの配列
infos	JSON obj	sku 固有のユーザー定義オブジェクト.
infos_poll	JSON obj	ポーリング処理による出力に付加される sku 固有のユーザー定義オブジェクト.
infos_od	JSON obj	オンデマンド処理にいる出力に付加される sku 固有のユーザー定義オブジェクト.

sku オブジェクト

10.1.3.7. bit オブジェクト

キー	データ型	説明
address	整数値又は文字列	レジスタの開始アドレス. 整数値又は '0x' で始まる 16 進数表記の文字列.
enable	論理値	このオブジェクトに基づくポーリングによるデータの読み込みを行うか否か. デフォルト値は true.
single	論理値	単一レジスタの書き込みに Modbus ファンクション 0x05 を用いる. false の場合は 0x0F を用いる. デフォルト値は true.
writeOnly	論理値	書き込み専用のレジスタ. デフォルト値は false.
bitsArray	JSON配列	bitsArray オブジェクトの配列 (最大 512 オブジェクト)

bit オブジェクト

10.1.3.8. register オブジェクト

キー	データ型	説明
address	整数値又は文字列	レジスタの開始アドレス. 整数値又は '0x' で始まる 16 進数表記の文字列.
enable	論理値	このオブジェクトに基づくポーリングによるデータの読み込みを行うか否か. デフォルト値は true.
single	論理値	単一レジスタの書き込みに Modbus ファンクション 0x06 を用いる. false の場合は 0x10 を用いる. デフォルト値は true.
writeOnly	論理値	書き込み専用のレジスタ. デフォルト値は false.
registersArray	JSON配列	registersArray オブジェクトの配列 (最大 512 オブジェクト)

register オブジェクト

10.1.3.9. rawRequest オブジェクト

キー	データ型	説明
key	整数値又は文字列	この RAW リクエストを呼び出すためのキー. (32byte)
unitId	整数値又は文字列	RAW リクエストに設定される Modbus ID. 整数値又は '0x' で始まる 16進数表記の文字列. デフォルト値は Modbus client に設定さる Modbus ID.
function	整数値又は文字列	RAW リクエストに設定される Modbus ファンクション番号. 整数値又は '0x' で始まる 16進数表記の文字列.
address	整数値又は文字列	RAW リクエストに設定されるレジスタのアドレス. 整数値又は '0x' で始まる 16進数表記の文字列.
value	整数値又は文字列	RAW リクエストに設定されるレジスタの値. 整数値又は '0x' で始まる 16進数表記の文字列.
form	文字列	出力メッセージの書式. 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'HEX'

rawRequest オブジェクト

10.1.3.10. bitsArray オブジェクト

キー	データ型	説明
include	論理値	こののビット値を送信するデータに含むか否か. デフォルト値は true.
form	文字列	このビット値の書式. 'bit'(1 or 0 の整数値) または 'bool'(true or false)、'BOOL'(TRUE or FALSE)、'hex'(16進表記の小文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'bit'.
key	文字列	このビット値に与えるJSONキー. (32byte)
array	整数値	配列表記としたい場合は、array キーにその素数の数を指定する.

bitsArray オブジェクト

10.1.3.11. registersArray オブジェクト

キー	データ型	説明
include	論理値	こののレジスタ値を送信するデータに含むか否か. デフォルト値は true.
form	文字列	このレジスタ値の書式. 'dec'(数値) または 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'dec'.
ratio	整数値	値の係数を10の階乗値で指定する. デフォルト値は 0.
key	文字列	このレジスタ値に与えるJSONキー. (32byte)
array	整数値	配列表記としたい場合は、array キーにその素数の数を指定する.

registersArray オブジェクト

10.1.4. モデリングファイル作成支援ツール

PD Handler Modbus 2 のパッケージには、モデリングファイルの構文を確認するツールと複数のモデリングファイルをマージする 2 つの作成支援ツールが用意されています。

パス名	説明
/usr/sbin/pd_handler_modbus_2_verify	モデリングファイル 構文確認ツール
/usr/sbin/pd_handler_modbus_2_merge	モデリングファイル マージツール

10.1.4.1. 構文確認ツール

pd_handler_modbus_2_verify は、作成したモデリングファイルの構文を確認するツールです。

pd_handler_modbus_2_verify コマンドのコマンドオプションを示します。

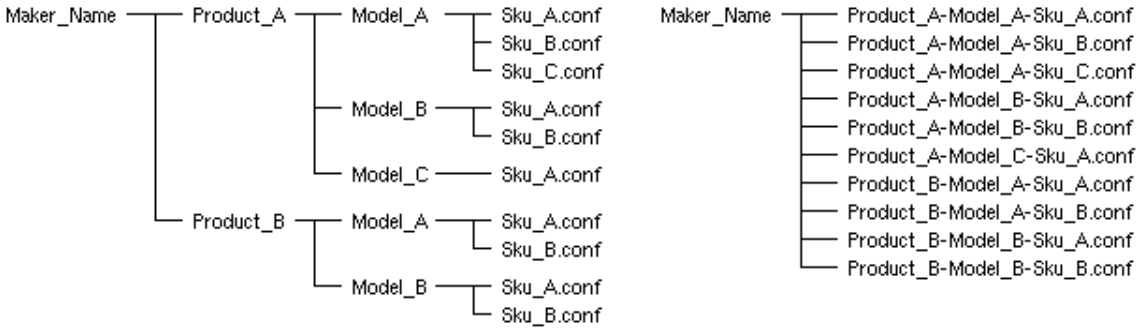
```
root# /usr/sbin/pd_handler_modbus_2_verify -h

usages: [-v] pd_handler_modbus_2_verify <model file>
  -v      Show progress.
```

pd_handler_modbus_2_verify コマンドの実行例を示します。

```
root# cd /usr/share/pdhms/modbus2/models/OMRON/ SmartPowerMonitor
root# /usr/sbin/pd_handler_modbus_2_verify -v KM-N1-FLK.conf
Read modeling file : KM-N1-FLK.conf
  Maker: OMRON
    Product: Smart Power Monitor
      Model: KM-N1-FLK
        SKU: 00
        SKU: 01
Verified total 1 makers done.
```

10.1.4.2. マージツール

PLC 機器によってはモデリングファイルが数百行に及ぶ場合もあり、全ての PLC 機器のモデリングを一つのファイルで管理することは現実的でないことから、モデリングファイルについては、プロダクト・モデルもしくは SKU 毎にファイルを分割し、必要に応じて下図のよにディレクトリ階層を細分化して管理することを推奨します。



ディレクトリ階層を細分化した例とフラット展開した例

マージツール pd_handler_modbus_2¥_merge は、指定されてたディレクトリ下に保存されている **.conf** のファイルディスクリプタを持つモデリングファイルを再回帰検索し、マージします。
ただし、ASCII 英数字と '-', '_', '.' 以外の記号文字を含むパス名には対応できません。

pd_handler_modbus_2_merge コマンドのコマンドオプションを示します。

```
root# /usr/sbin/pd_handler_modbus_2_merge -h

usages: pd_handler_modbus_2_merge [-v][-o <file>][<root path>]
  -v              Show progress.
  -o file       Output file. [default: stdout]
  root path    Root path. [default: ./]
```

pd_handler_modbus_2_merge コマンドの実行例を示します。

```
root# cd /usr/share/pdhms/modbus2/models/
root# /usr/sbin/pd_handler_modbus_2_merge -v -o /tmp/foo.conf WatanabeElectricInc/
Found 14 files in WatanabeElectricInc/.
Load file: WatanabeElectricInc/WMS/PE1N/00A000_SW4x.conf
  Maker: Watanabe Electric Inc.
    Product: WMS
      Model: PE1N
        SKU: 00A000_SW4x
Load file: WatanabeElectricInc/WMS/PE1N/00A000.conf
  Maker: Watanabe Electric Inc.
    Product: WMS
      Model: PE1N
        SKU: 00A000
          :
        (中略)
          :
Merge 1 makers.
  Maker: Watanabe Electric Inc.
    Product: WMS
      Model: PE1N
        SKU: 00A000_SW4x
        SKU: 00A000
      Model: PE6N
        SKU: 00A007_SW4x-A
        SKU: 00A007_SW4x-B
        SKU: 00A007-A
        SKU: 00A007-B
    Product: WMB
      Model: DIO8R
        SKU: 00D000
        SKU: 00D000_SW4x_DIO
      Model: DI16A
        SKU: 00D000
      Model: AI8
        SKU: 36D000
      Model: DI16
        SKU: 00D000
      Model: DIO8RA
        SKU: 00D000
        SKU: 00D000_SW4x_DIO
      Model: MAI6
        SKU: 36FD000
Dump to /tmp/foo.conf.
```

10.2. PD Handler Modbus 2 Client

PD Handler Modbus 2 Clien は、モデリングファイル pd_handler_modbus_2_model.conf と設定ファイル pd_handler_modbus_2_client.conf、CSV ファイル pd_handler_modbus_client.csv の設定に基づき定期的に PLC 機器に Modbus プロトコルで接続しデータを読み取ります。
また、PD Repeater を介しクラウドから送られる JSON 文字列に基づき PLC 機器に Modbus プロトコルで接続しデータを読み書きすることもできます。

10.2.1. デフォルトパス

PD Handler Modbus 2 Clien に関連するファイルのデフォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_handler_modbus_2_client	常駐実行オブジェクト（デーモン）
/usr/sbin/pd_handler_modbus_2_cli	コマンドラインインタフェース
/lib/systemd/system/pd_handler_modbus_2_client.service	Systemd Service ファイル
/etc/init.d/pd_handler_modbus_2_client	RC ファイル
/var/webui/config/pd_handler_modbus_2_client.conf	設定ファイル
/var/webui/config/pd_handler_modbus_2_client_custom.conf	カスタマイズ用設定ファイル
/var/webui/config/pd_handler_modbus_2_model.conf	Modbus モデリングファイル
/var/webui/config/pd_handler_modbus_2_model.conf.default	Modbus モデリングファイル(デフォルト)
/var/webui/upload_dir/ pd_handler_modbus_2_client.csv	CSV ファイル
/var/run/pd_handler_modbus_2_client.pid	PID ファイル

PD Handler Modbus 2 Clien に関連するファイルのデフォルトパス

Modbus モデリングファイル(デフォルト)は、Modbus モデリングファイルが存在しない場合に読み込まれます。

10.2.2. 設定ファイルの書式

10.2.2.1.構文

PD Handler Modbu Client 2 の設定ファイルは次の書式で構成されています。

```
{
    "custom_file": "カスタマイズ用設定ファイル",
    "csv_file": "CSV ファイル",
    "model_file": "モデリングファイル",
    "clients" : [
        {
            client オブジェクト
        },
        {
            client オブジェクト
        },
    ]
}
```

10.2.2.2. ルートオブジェクト

キー	データ型	説明
custom_file	文字列	カスタマイズ用設定ファイルのパス名．デフォルト値は、'/var/webui/config/pd_handler_modbus_2_client_custom.conf'. (MAXPATHLEN)
CSV_file	文字列	CSVファイルののパス名．デフォルト値は、'/var/webui/upload_dir/pd_handler_modbus_2_client.csv'. (MAXPATHLEN)
model_file	文字列	Modbusモデリングファイルのパス名．デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
clients	JSON配列	clientsオブジェクトの配列（最大 64 オブジェクト）

ルートオブジェクト

10.2.2.3. clients オブジェクト

キー	対象	データ型	説明
enable	共通	論理値	デフォルト値はfalse
localname		文字列	デバイスのローカル名(デバイス番号). (32byte)
bind		文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to		文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size		整数値	データのバッファサイズ(byte). デフォルト値は4096
receive		論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse.
reply		論理値	下流方向制御に対しステータスメッセージを返すか否か. デフォルト値は false.
protocol		文字列	接続プロトコル'tcp' 又は'rtu' を指定. デフォルト値は'tcp'.
timeout		整数値	データを取得出来ない場合のタイムアウト(msec). デフォルト値は5000.
id_form		文字列	Unit Id(Modbus Id) の表記フォーマト. 'dec'(数値) または 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'dec'.
timestamp_key		文字列	タイムスタンプのキーを設定する. デフォルト値は 'timestamp' (32byte)
infos		JSON obj	出力に付加される Modbus クライアント ノード固有のユーザー定義オブジェクト.
node	TCP	文字列	TCP 接続 PLC 機器の IP アドレス. デフォルト値は'127.0.0.1'
port		整数値	TCP 接続 PLC 機器のポート番号. デフォルト値は 502
device	RTU	文字列	シリアル接続PLC 機器のデバイス名. デフォルト値は '/dev/tty00'
rtu_speed		整数値	シリアル接続 PLC 機器のビットレート. デフォルト値は 115200
rtu_bits		整数値	シリアル接続 PLC 機器のビット数. 8 又は 7. デフォルト値は 8.
rtu_parity		文字列	シリアル接続PLC 機器のパリティ. 'none','even','odd' のいずれか. デフォルト値は'none'.
rtu_stop		整数値	シリアル接続PLC 機器のストップビット. 1 又は2. デフォルト値は1.
rtu_rts_delay		整数値	シリアル接続PLC 機器のRTS ディレイ値(usec). 0 の場合は自動設定される値.
rtu_reset		整数値	設定される値を越えて接続エラーが連続して発生した場合にシリアルポートをリセットする. デフォルト値は 10.
acquisitions	共通	JSON 配列	acquisitions オブジェクト

clients オブジェクト

10.2.2.4. acquisitions オブジェクト

キー	データ型	説明
enable	論理値	デフォルト値は false.
makre	文字列	モデルファイルに定義されるメーカー名 (64byte)
product	文字列	モデルファイルに定義されるプロダクト名 (64byte)
model	文字列	モデルファイルに定義されるモデル名 (64byte)
sku	文字列	モデルファイルに定義される sku 名 (64byte)
unitId	文字列又は整数	PLC 機器のModbus ユニットID. 1 ～ 247 又は255 (TCP プロトコルのみ). '0x' で始まる 16進数表記の文字列入力も可能.
include	論理値	unitId を送信するデータに含むか否か. デフォルト値は true.
interval	整数値	データを取得する間隔(sec). デフォルト値は 60.
time_sync	論理値	基準時刻制御モード. デフォルト値は false
base_time	文字列	基準時刻制御モードの基準時刻を 'HH:MM' 形式で指定する. デフォルト値は '00:00'
infos	JSON obj	出力に付加されるデータ取得対象 PLC 機器固有のユーザー定義オブジェクト.

acquisitions オブジェクト

10.2.3. カスタマイズ用設定ファイルの書式

カスタマイズ用設定ファイルは、Web UI の出力する設定ファイルに対し、cvs_file, model_file, bind, push_to, buffer_size, id_form, infos の各値をカスタマイズ(オーバーライト)します。

10.2.3.1. 構文

```
{
    "csv_file": "CSV ファイル",
    "model_file": "モデリングファイル",
    "clients" : [
        {
            カスタマイズ用 clients オブジェクト,
        },
        {
            カスタマイズ用 clients オブジェクト,
        }
    ]
}
```

10.2.3.2. カスタマイズ用ルートオブジェクト

キー	データ型	説明
CSV_file	文字列	CSVファイルののパス名. デフォルト値は、'/var/webui/upload_dir/pd_handler_modbus_2_client.csv'. (MAXPATHLEN)
model_file	文字列	Modbusモデリングファイルのパス名. デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
clients	JSON配列	カスタマイズ用 clients オブジェクトの配列 (最大 64 オブジェクト)

カスタマイズ用ルートオブジェクト

10.2.3.3. カスタマイズ用 clients オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
bind	文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte). デフォルト値は4096
id_form	文字列	Unit Id(Modbus Id) の表記フォーマット. 'dec'(数値) または 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'dec'.
infos	JSON obj	出力に付加される Modbus クライアント ノード固有のユーザー定義オブジェクト.

カスタマイズ用 clients オブジェクト

10.2.4. CSV ファイル

PD Handler Modbus Client 2 は、CSV ファイル /var/webui/upload_dir/pd_handler_modbus_2_client.csv が存在すると同ファイルより acquisitions オブジェクトに相当する設定パラメータを読み込み、設定ファイル /var/webui/config/pd_handler_2_modbus.conf から読み込んだパラメータを上書きします。
CSV ファイルに設定される情報は、clients オブジェクトの localname と acquisitions オブジェクトの unitId, include, maker, product, model, sku, infos, interval, time_sync, base_time です。　ただし、infos オブジェクトは、2 組の key と値(文字列)に限定され、また不要な場合は空欄とします。
パラメータの並びは次の通りで、パラメータの区切りはコンマ ','、先頭がシャープ '¥#' 又はスラッシュ '/' の場合はコメント行として扱います。

localname, include, maker, product, model, sku, key1, value1, key2, value2,
interval, time_sync, base_time

パラメータのデータ型は acquisitions オブジェクトと同じですが、JSON 文字列とは異なり文字列に空白が含まれない限り、文字列を "" で括る必要はありません。
なお、localname が 設定ファイルに存在しない場合は、無効となります。

```
#localname, unitId, include, maker, product, model, sku, ¥
    key1, val1, key2, val2, interval, time_sync, base_time
mdc2dev_0000001, 1, 1, "Toukou Toushiba", "SmaMe Series", S2MS-RNS22, 0001, ¥
    IBS, 1-1, Mater, 1, 300, true, 00:00 # comment
mdc2dev_0000001, 2, 1, "Toukou Toushiba", "SmaMe Series", S2MS-RNS22, 0001, ¥
    IBS, 1-1, Mater, 2, 300, true, 00:00 # comment
mdc2dev_0000001, 3, 1, "Toukou Toushiba", "SmaMe Series", S2MS-RNS22, 0001, ¥
    IBS, 1-2, Mater, 1, 300, true, 00:00 # comment
mdc2dev_0000001, 4, 1, "Toukou Toushiba", "SmaMe Series", S2MS-RNS22, 0001, ¥
    IBS, 1-2, Mater, 2, 300, true, 00:00 # comment
```

10.2.5. 基準時刻制御モード

基準時刻制御は、特定の時刻にデータを取得する機能です。
clients オブジェクトの time_sysnc キーを true に設定し、interval キーと base_time キーで取得間隔と取得時刻を設定します。
基準時刻制御における「取得時間間隔」は 60, 120, 180, 300, 600, 900, 1800, 3600, 7200, 10800, 14400, 21600, 28800, 43200 と 86400 の倍数に限られます。「取得時間間隔」としてこれら以外の値が設定されると PD Handler Modbus 2 Client 内で次のように扱われます。

取得時間間隔の設定値	実動作値
0 ～ 119	60
120 ～ 179	120
180 ～ 299	180
300 ～ 599	300
600 ～ 899	600
900 ～ 1799	900
1800 ～ 3599	1800
3600 ～ 7199	3600
7200 ～ 10799	7200
10800 ～ 14399	10800
14400 ～ 21599	14400
21600 ～ 28799	21600
28800 ～ 43199	28800
43200 ～ 86399	43200
86400～	86400 の倍数

基準時刻制御における取得時間間隔の設定と実動作値

「基準時刻」とは動作の起点となる時刻で、例えば「取得時間間隔」を 300 とし、「基準時刻」を”00:01”とした場合、データの取得は 00:01, 00:06, 00:11 ... 00:56, 01:01 ... 23:56, 00:01 の定刻に行われます。
データの取得開始時刻は「基準時刻」に設定さた時刻そのものではなく、「基準時刻」と「取得時間間隔」から算定される直近の時刻となります。
例えば 08:30 に「基準時刻」”01:05”、 「取得時間間隔」10800 の設定が行われた場合、最初のデータ取得は 10:05 に行われ、以降 13:05, 16:05, 19:05, 22:05, 01:05 の順におこなわれます。

10.2.6. メッセージオブジェクト

入出力メッセージに用いられる JSON オブジェクトを示します。

キー	内容	データ型	方向	備考
timestamp	データ取得日時	文字列	出力	ISO8601 拡張書式
unitId	Modbus Unit ID	整数値又は文字列	入出力	clients オブジェクト の 'include' が true 時のみ出力.
makre	メーカー名	文字列	出力	モデリングファイルの maker オブジェクトの 'include' が true 時のみ.
product	プロダクト名	文字列	出力	モデリングファイルの product オブジェクトの 'include' が true 時のみ.
model	モデル名	文字列	出力	モデリングファイルの model オブジェクトの 'include' が true 時のみ.
sku	sku 名	文字列	出力	モデリングファイルの sku オブジェクトの 'include' が true 時のみ.
reply_to	リクエストメッセージの MD5	文字列	出力	clients オブジェクトの 'reply' が true 時のみ.
request_from	リクエストメッセージの送付元クラウド ID	文字列	出力	clients オブジェクトの 'reply' が true 時のみ.
result	制御ステータス	論理値	出力	clients オブジェクトの 'reply' が true 時のみ.
reason	エラーの理由	文字列	出力	clients オブジェクトの 'reply' が true 時のみ.
write	書き込み制御	JSON obj	入力	モデリングファイルに定義されるキーと値
rawRequest	RAW リクエスト	JSON obj	入力	モデリングファイルに定義される RAW リクエストキーと値. モデリングファイルに定義される値を持ちいる場合は値を null にする.

PD Handler Modbus2 Client のメッセージオブジェクト

10.2.7. クラウドからの制御

クラウドから PD Repeater を介し制御(JSON)文字列を送ることで、接続されている PLC 機器のレジスタを読み書きすることができます。
制御文字列には、PLC 機器を特定するため acquisitions オブジェクトの unitId を記載します。

例えば、Modbus ID 3 の PLC 機器のレジスタを読み込むのであれば、その制御文字列は次のようになります。

```
{
    "unitId”: 3
}
```

これに対する、応答メッセージは、次のようになります。

```
{
    "timestamp": "2019-11-07T11:19:08.949+09:00"
    "request_from":"0x13(0)",
    "reply_to":"6bee94d5a2f29eb63b5faecd7e8af8d9", "result": true,
    "unitId": 3,
    "doValue_ch1": 0, "doValue_ch2": 0, "doValue_ch3": 0, "doValue_ch4": 0,
    "diValue": [0, 0, 0, 0],
    "pulseCoutLimit": [99999999, 99999999, 99999999, 99999999],
    "pulseCountReset_ch1": 0, "pulseCountReset_ch2": 0,
    "pulseCountReset_ch3": 0, "pulseCountReset_ch4": 0,
    "pulseCount": [10000, 0, 0, 0]
}
```

ここで、reply_to は、制御文字列の MD5 です。

同 PLC 機器の出力レジスタに値を書き込むのであれば、その制御文字列は次のようになります。

```
{
    "unitId": 3, "write":{"doValue_ch1": 1, "pulseCountReset_ch1": 1}
}
```

ここで write オブジェクトに与えられるキーと値は、モデリングの outputBits オブジェクト配列内の bit オブジェクト、又は outputRegisters オブジェクト配列内の registers オブジェクトに指定される 'key', 'type', 'ratio', 'array' に応じた値でなくてはなりません。

これに対する、応答メッセージは、次のようになります。


```
{
    "timestamp":"2019-06-11T15:30:05.758+09:00",
    "request_from":"0x13(0)",
    "reply_to":"38bfe79195fc09b37b6a85b6fdc97d73", "result": true,
    "doValue_ch1": 1, "doValue_ch2": 0, "doValue_ch3": 0, "doValue_ch4": 0,
    "pulseCountReset_ch1": 1
}
```


書込み制御は、outputBits オブジェクト配列内の bit オブジェクト、又は outputRegisters オブジェクト配列内の registers オブジェクトの単位で行われます。

書込み処理においては、連続しない複数のレジスタに対する制御を効率良く処理するため、bit オブジェクト又は register オブジェクトに定義されるレジスタの全てを PLC 機器からプログラムに読み込み、write キーに指定されるレジスターのみを書き換えた後、レジスタの全てを PLC 機器へ書き戻します。

10.2.8. CLI (pd_handler_modbus_2_cli)

pd_handler_modbus_2_cli は、クラウドからの制御をローカルに行うためのアプリケーションです。
PLC 機器の初期設定(ディジタル出力もしくは出力レジスタへの書き込み)やモデリングファイルの確認などに用いるものですが、制御対象が RTU 接続の場合は、
pd_handler_modbus_2_client とインタフェースが競合するため、
pd_handler_modbus_2_client を停止した上で使用することを推奨します。

pd_handler_modbus_2¥_cli のオプションを示します。

```
root# /usr/sbin/pd_handler_modbus_2_cli -h

usages: pd_handler_modbus_2_cli [-adqr][-c <conf file>][-C <CSV file>][-l <local name>]
                                [-m <model file>] <control JSON strings>
  -a              Force enable all for read.
  -c conf file    Conf file. [default: /var/webui/config/pd_handler_modbus_2_client.conf]
  -C CSV file     CSV file for multiple acquisitions.
                                [default: /var/webui/config/pd_handler_modbus_2_client.csv]
  -d              Debug mode.
  -l local name   Local name. [default: mdcdev_0000001]
  -m model file   Modbus model file.
                                [default: /var/webui/config/pd_handler_modbus_2_model.conf]
  -q              Quiet mode.
  -r              Read only for writ process.
```

'-a' オプションは、モデリング上 disable ('enable' キーが false) とされているレジスタも読み出します。

例えば、Modbus ID 3 の PLC 機器のレジスタを読み込むのであれば、
pd_handler_modbus_2_cli の引数にクラウドからの制御と同じ制御文字列を与えます。

```
root# pd_handler_modbus_2_cli '{"unitId": 3}'
```

これに対し応答メッセージは、標準出力に返されます。

```
{
    "timestamp": "2019-06-11T15:30:05.758+09:00",
    "maker":"Toukou Toushiba", "product":"SmaMe Series", "model":"S2MS-RNS22", "sku":"0001",
    "switch": 1, "power": 1234, "reversePower": 4567, "current":[2.5, 3.1],
    "voltage":[101.2, 101.3], "instPower": 5678, "serialNumber":112345678
}
```

クラウドから制御した場合と異なり、reply_to と result は出力されません。

同 PLC 機器の出力レジスタに値を書き込むのであれば、pd_handler_modbus_2_cli の引数は次のようになります。

```
# pd_handler_modbus_2_cli '{"unitId": 3, "write":{"switch": 0}}'
```

ここで write オブジェクトに与えられるキーと値は、モデリングの outputBits オブジェクト配列内の bit オブジェクト、又は outputRegisters オブジェクト配列内の registers オブジェクトに指定される 'key', 'type', 'ratio', 'array' に応じた値でなくてはなりません。

これに対する応答メッセージは、次のようになります。


```
{
    "timestamp":"2019-06-11T15:30:05.758+09:00",
    "maker":"Toukou Toushiba", "product":"SmaMe Series", "model":"S2MS-RNS22", "sku":"0001",
    "switch": 0
}
```

10.3. PD Handler Modbus 2 Server

PD Handler Modbus Server 2 は、モデリングファイル pd_handler_modbus_2_model.conf と設定ファイル pd_handler_modbus_2_server.conf の設定に基づき共有メモリにレジスタマップを構成し PLC 機器(Modbus クライアント)からの接続を待ち受け、PLC 機器(Modbus クライアント)の書き込み操作によりレジスタの値を更新すると共に、更新されたレジスタとその値を PD Repeater を介してクラウドへ送ります。
また、PD Repeater を介しクラウドから送られる JSON 文字列に基づき、レジスタマップを読み書きすることができます。
レジスタマップは、60 秒毎に更新があれば、共有メモリからレジスタマップファイルに出力（バックアップ）されます。

10.3.1.デフォルトパス

PD Handler Modbus 2 Server に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_handler_modbus_2_server	常駐実行オブジェクト（デーモン）
/etc/init.d/pd_handler_modbus_2_server	RC ファイル
/lib/systemd/system/pd_handler_modbus_2_server.service	Systemd Service ファイル
/var/webui/config/pd_handler_modbus_2_server.conf	設定ファイル
/var/webui/config/pd_handler_modbus_2_server_custom.conf	カスタマイズ用設定ファイル
/var/webui/config/pd_handler_modbus_2_model.conf	Modbus モデリングファイル
/var/webui/config/pd_handler_modbus_2_model.conf.default	Modbus モデリングファイル(デフォルト)
/var/webui/.modbus_2_server/<デバイス番号>.map	レジスタマップファイル
/var/run/pd_handler_modbus_2_server.pid	PID ファイル

PD Handler Modbus 2 Server に関連するファイルのデォルトパス

Modbus モデリングファイル(デフォルト)は、Modbus モデリングファイルが存在しない場合に読み込まれます。

10.3.2. 設定ファイルの書式

10.3.2.1. 構文

```
{
    "custom_file": "カスタマイズ用設定ファイル",
    "model_file": "モデリングファイル",
    "backup_interval":  バックアップ周期,
    "servers" : [
        {
            servers オブジェクト
        },
        {
            servers オブジェクト
        }
     ]
}
```

10.3.2.2. ルートオブジェクト

キー	データ型	説明
custom_file	文字列	カスタマイズ用設定ファイルのパス名. デフォルト値は、'/var/webui/config/pd_handler_modbus_2_server_custom.conf'. (MAXPATHLEN)
model_file	文字列	Modbusモデリングファイルのパス名. デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
backup_interval	整数値	レジスタをファイルにバックアップする周期(sec)を指定する. デフォルト値は 60.
servers	JSON配列	serversオブジェクトの配列 (最大 8 オブジェクト)

ルートオブジェクト

10.3.2.3. servers オブジェクト

server オブジェクトは、PD Handler Modbus 2 Serverの動作を規定する設定オブジェクトです。server オブジェクトの配列数は、最大8 個. Modbus のプロトコル(TCP, RTC) に依存するオブジェクトとプロトコルに依存しない共通のオブジェクトがあります。

キー	対象	データ型	説明
enable	共通	論理値	デフォルト値はfalse
localname		文字列	デバイスのローカル名(デバイス番号). (32byte)
map_file		文字列	レジスタマップファイルのパス名. デフォルト値は /var/webui/.modbus_2_server/<デバイス番号>.map
bind		文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/＜デバイス番号＞.sock が設定されます.
push_to		文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
buffer_size		整数値	データのバッファサイズ(byte). デフォルト値は4096
receive		論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse.
reply		論理値	下流方向制御に対しステータスメッセージを返すか否か. デフォルト値は false.
protocol		文字列	接続プロトコル'tcp' 又は'rtu' を指定. デフォルト値は'tcp'.
timeout		整数値	データを取得出来ない場合のタイムアウト(msec). デフォルト値は5000.
maker		文字列	モデルファイルに定義されるメーカー名 (64byte).
product		文字列	モデルファイルに定義されるプロダクト名 (64byte)
model		文字列	モデルファイルに定義されるmodel名 (64byte)
sku		文字列	モデルファイルに定義される sku 名 (64byte)
unitId		整数又は文字列	PLC機器の Modbus ユニット ID. 1 ～ 247 又は 255 (TCPプロトコルのみ). '0x' で始まる 16進数表記の文字列入力も可能.
id_form		文字列	Modbus Id の表記フォーマット. 'dec'(数値) または 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'dec'.
timestamp_key		文字列	タイムスタンプのキーを設定する. デフォルト値は 'timestamp' (32byte)
include		論理値	unitId を送信するデータに含むか否か. デフォルト値は true.
infos		JSON obj	出力に付加される Modbus サーバ ノード固有のユーザー定義オブジェクト.
node	TCP	文字列	TCP 接続待ち受け IP アドレス. デフォルト値は'127.0.0.1'
port		整数値	TCP 接続待ち受けポート番号. デフォルト値は 502
device	RTU	文字列	シリアル接続のデバイス名. デフォルト値は'/dev/tty00'
rtu_speed		整数値	シリアル接続のビットレート. デフォルト値は 115200
rtu_bits		整数値	シリアル接続のビット数. 8 又は 7. デフォルト値は 8.
rtu_parity		文字列	シリアル接続のパリティ. 'none','even','odd' のいずれか. デフォルト値は'none'.
rtu_stop		整数値	シリアル接続のストップビット. 1 又は2. デフォルト値は1.
rtu_rts_delay		整数値	シリアル接続のRTS ディレイ値(usec). 0 の場合は自動設定される値.

10.3.3. カスタマイズ用設定ファイルの書式

カスタマイズ用設定ファイルは、Web UI の出力する設定ファイルに対し、model_file , bind, push_to, buffer_size, id_form, infos の各値をカスタマイズ(オーバーライト)します。

10.3.3.1. 構文

```
{
    "model_file": "モデリングファイル",
    "clients" : [
        {
            カスタマイズ用 servers オブジェクト,
        },
        {
            カスタマイズ用 servers オブジェクト,
        }
    ]
}
```

10.3.3.2. カスタマイズ用ルートオブジェクト

キー	データ型	説明
model_file	文字列	Modbusモデリングファイルのパス名. デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
servers	JSON配列	カスタマイズ用 servers オブジェクトの配列 (最大 8 オブジェクト)

カスタマイズ用ルートオブジェクト

10.3.3.3. カスタマイズ用 servers オブジェクト

キー	データ型	説明
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
bind	文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte). デフォルト値は4096
id_form	文字列	Unit Id(Modbus Id) の表記フォーマト. 'dec'(数値) または 'hex'(16進表記の文字列)、'HEX'(16進表記の大文字列). デフォルト値は 'dec'.
infos	JSON obj	出力に付加される Modbus サーバー ノード固有のユーザー定義オブジェクト.

カスタマイズ用 serves オブジェクト

10.3.4. メッセージオブジェクト

入出力メッセージに用いられる JSON オブジェクトを示します。

キー	内容	データ型	方向	備考
timestamp	データ取得日時	文字列	出力	ISO8601 拡張書式
unitId	Modbus Unit ID	整数値又は文字列	出力	servers オブジェクト の 'include' が true 時のみ出力.
makre	メーカー名	文字列	出力	モデリングファイルの maker オブジェクトの 'include' が true 時のみ.
product	プロダクト名	文字列	出力	モデリングファイルの product オブジェクトの 'include' が true 時のみ.
model	モデル名	文字列	出力	モデリングファイルの model オブジェクトの 'include' が true 時のみ.
sku	sku 名	文字列	出力	モデリングファイルの sku オブジェクトの 'include' が true 時のみ.
reply_to	リクエストメッセージの MD5	文字列	出力	servers オブジェクトの 'reply' が true 時のみ.
request_from	リクエストメッセージの送付元クラウド ID	文字列	出力	servers オブジェクトの 'reply' が true 時のみ.
result	制御ステータス	論理値	出力	servers オブジェクトの 'reply' が true 時のみ.
reason	エラーの理由	文字列	出力	servers オブジェクトの 'reply' が true 時のみ.
write	書き込み制御	JSON obj	入力	モデリングファイルに定義されるキーと値

PD Handler Modbus 2 Servers のメッセージオブジェクト

10.3.5. クラウドからの制御

クラウドから PD Repeater を介し制御(JSON)文字列を送ることで、レジスタマップの読み取りと入力レジスタに対する書き込みができます。

レジスタマップを読み取る場合は、空の制御文字を送ります。

```
{ }
```

これに対する、応答メッセージは、次のようになります。

```
{
    "timestamp":"2019-06-11T15:30:05.758+09:00",
    "request_from":"0x13(0)",
    "reply_to":"99914b932bd37a50b983c5e7c90ae93b","result": true,
    "unitId": 16,
    "maker":"Plat Home", "product":"Modbus Handler", "model":"Test Server", "sku":"001",
    "do00": "0x24", "do01": "0x36", "do02": "0xcd", "do03": "0xac",
    "di00": "0x00", "di01": "0x00", "di02": "0x00", "di03": "0x00",
    "rego00": 128, "regi01": 25, "regi02": -1, "regi03": 45,
    "regi": [0, 0, 0, 0]
}
```

ここで、reply_to は、制御文字列の MD5 です。

入力レジスタに値を書き込むのであれば、その制御文字列は次のようになります。

```
{
    "write": {"di00": "0xff", "regi": [123, 456, 678, 910]}
}
```

ここで write オブジェクトに与えられるキーと値は、モデリングの inputBits オブジェクト配列内の bit オブジェクト、又は inputRegisters オブジェクト配列内の registers オブジェクトに指定される 'key', 'type', 'ratio', 'array' に応じた値でなくてはなりません。

これに対する、応答メッセージは、次のようになります。

```
{
    "timestamp":"2017-06-11T16:35:13.653+09:00",
    "reply_to":"9e5d68f8ce9bb5dc3e86f2d3c2ef4167","result": true,
    "request_from":"0x13(0)",
    "unitId": 16,
    "maker":"Plat Home", "product":"Modbus Handler", "model":"Test Server", "sku":"001",
    "di00": "0xff", "regi": [123, 456, 678, 910]
}
```

11. PD Handler SW4x

PD Handler SW4x は、セイコーインスツル社製 SW4000 シリーズ無線センサーネットワークにおいて、同シリーズのシリアル接続型ベースデバイスを介し、設定ファイル pd_handler_sw4x.conf とモデリングファイル pd_handler_sw4x_model.conf、RTU モデリングファイル pd_handler_sw4x_rtu.conf、並びに Modbus 用モデリングファイル pd_handler_modbus_2_model.conf の設定に基づき、同ネットワークに接続されるセンサーノードもしくはモニターノードより送信されるデータをハンドリングします。
また、PD Repeater を介しクラウドから送られる JSON 文字列によるセンサーノードもしくはモニターノードに対する制御も可能です。

11.1. 対応ノード一覧

PD Handler SW4x がサポートするノードの一覧を示します。

名称	型番	Unit Type	機能
温度ノード	SW-42P0-1x01	0x00	温度のモニター
温・湿度ノード	SW-4210-1202	0x01	温度・湿度のモニター
照度ノード	SW-4210-1205	0x02	照度のモニター
温・湿度・照度ノード	SW-4210-1204	0x03	温度・湿度・照度のモニター
人感ノード(活動量)	SW-4220-1000	0x09	人の活動量をモニター
パルスカウントノード	SW-4240-1000	0x0A	入力パルスの積算カウント
人感ノード（人感)	SW-4220-1010	0x0B	人を検出時に送信
リモコン温・湿度ノード	SW-42J0-1202	0x0D	温度・湿度のモニターと赤外線リモコン信号の発信
パルスピックノード	SW-42K0-1000	0x0F	入力パルスの積算カウント
電流センサーノード(5A)	SW-42DD-1000	0x10	電流・電力・積算電力のモニター
電流センサーノード(200A)	SW-42DD-1100	0x11	
電流センサーノード	SW-42D0-1000	0x12	
CO2 ノード	SW-4230-1000	0x20	二酸化炭素濃度のモニター
電力モニターノード	SW-4260-1110	0x21	オムロン社製　KM20-B40-FLK、KM50-C・E シリーズのデータ送信
Modbus RTU ノード	SW-4280-1000	0x23	Modbus RTU 機器の制御
電力モニターノード	SW-4260-1120	0x28	オムロン社製　KM-N1-FLK シリーズのデータ送信

PD Handler SW4x がサポートするノードの一覧

11.2.デフォルトパス

PD Handler SW4x に関連するファイルのデォルトパスは次の通りです。

パス名	説明
/usr/sbin/pd_handler_sw4x	常駐実行オブジェクト（デーモン）
/usr/sbin/pd_handler_sw4x_cli	コマンドラインインタフェース
/etc/init.d/pd_handler_sw4x	RC ファイル
/lib/systemd/system/pd_handler_sw4x.service	Systemd Service ファイル
/var/webui/config/pd_handler_sw4x.conf	設定ファイル
/var/webui/config/pd_handler_sw4x_custom.conf	カスタマイズ用設定ファイル
/var/webui/config/pd_handler_sw4x_model.conf	モデリングファイル
/var/webui/config/pd_handler_sw4x_rtu.conf	RTU モデリングファイル
/var/webui/config/pd_handler_modbus_2_model.conf	Modbus モデリングファイル
/var/webui/config/pd_handler_modbus_2_model.conf.default	Modbus モデリングファイル(デフォルト)
/var/run/pd_handler_sw4x.pid	PID ファイル

PD Handler SW4x に関連するファイルのデォルトパス

PD Handler SW4x のモデリングは実行オブジェクトにハードコーディングされているため、モデリングファイルは、出力メッセージの書式を変更する場合にのみ必要となります。変更する場合は /usr/share/pdhms/sw4x/pd_handler_sw4x_model.conf.default を /var/webui/config/pd_handler_sw4x_model.conf へコピーし、値のみを編集して下さい。配列の要素数とその順列が変更された場合の動作は保証されません。

11.3. モデリングファイルの書式

11.3.1. 構文

```
{
    "モデルキー": {
            モデルオブジェクト
    }
    "モデルキー": {
            モデルオブジェクト
    }
}
```

11.3.2. モデルキー

モデルキーの一覧を示します。

キー	名称	型番
SW4X_COMMON	モデルに依存しない共通項の設定用	
SW42P0_1x01	温度ノード	SW-42P0-1x01
SW4210_1202	温・湿度ノード	SW-4210-1202
SW4210_1205	照度ノード	SW-4210-1205
SW4210_1204	温・湿度・照度ノード	SW-4210-1204
SW4220_1000	人感ノード(活動量)	SW-4220-1000
SW4240_1000	パルスカウントノード	SW-4240-1000
SW4220_1020	人感ノード (人感)	SW-4220-1010
SW42J0_1202	リモコン温・湿度ノード	SW-42J0-1202
SW42K0_1000	パルスピックノード	SW-42K0-1000
SW42DD_1000	電流センサーノード(5A)	SW-42DD-1000
SW42DD_1100	電流センサーノード(200A)	SW-42DD-1100
SW42D0_1000	電流センサーノード	SW-42D0-1000
SW4230_1000	CO2 ノード	SW-4230-1000
SW4260_1x10	電力モニターノード	SW-4260-1110
SW4280_1000	Modbus RTU ノード	SW-4280-1000
SW4260_1x20	電力モニターノード	SW-4260-1120
SW4100_1000	ルーターノード	SW-4100-1000
SB4020_1000	OpenBlocks IoT Family アドオンモジュール	SB-4020-1000
SW4000_1000	USB ベース	SW-4000-1000
SW4300_1000	Ethernet ベース	SW-4300-1000
SW4500_1000	RS-232 ベース	SW-4500-1000

モデルキーの一覧

11.3.3. モデルオブジェクト

モデルオブジェクトは、次のオブジェクトから構成されています。　但し、enable, nodeInfos, predictio, inits, polls の各キーはモデルによって存在しない場合があります。

キー	データ型	説明
enable	論理型	モデルのデータをハンドリングするか否か. デフォルト値は true.
refuse_empty	論理型	include されているデータが存在しない場合、unitType や timestamp 等の共通項の送信を行わない. デフォルト値は false.
keys	JSON配列	モデルの持つデーターモデルの並び.
infos	JSON obj	出力に付加されるモデルのユーザー定義オブジェクト.
precision	整数値	JSON文字列にダンプされる際のリアル型データの精度(libjansson に与えるパラメータ).
nodeInfos	JSON obj	接続されているノード(RID)の一覧.
inits	JSON配列	PD Handler SW4x 起動時にノードに与えられる動作パラメータ設定コマンドの並び.
polls	JSON配列	定期データ取得用オブジェクトの並び.

モデルオブジェクト

11.3.4. keys オブジェクト

keys オブジェクトの数(配列の要素数)と設定可能なキーはモデルと配列の要素のデータの型等によって異なります。
配列の要素とその順列は、モデル毎に固定されているため、値を変更する場合は /usr/share/pdhms/sw4x/pd_handler_sw4x_model.conf.default を /var/webui/config/pd_handler_sw4x_model.conf へコピーし、値のみを編集して下さい。
配列の要素数とその順列が変更された場合の動作は保証されません。

キー	データ型	説明
key	文字列	キーの文字列 (32byte).
include	論理型	このキーを出力に含めるか否か.
form	文字列	値のフォーマットとして 'bool'(論理表記 true/false),'dec'(10進数),'HEX'(16進数大文字),'hex'(16進数小文字)のいずれかを指定する.
ratio	整数値	値の係数を10の階乗値で指定する.

keys オブジェクト

11.3.5. nodeInfos オブジェクト

nodeInfos オブジェクトは、接続されているノードど一覧を示す付加情報であり、SW4X_COMMON オブジェクトにのみ記載されます。
nodeInfos を出力に含めるか否かは、keys オブジェクト配列の 18 番目の要素に指定します。

キー	データ型	説明
ridNNN	文字列	ridNNN & 文字列 & RID 'NNN' に接続されるノードを示す任意の文字列 (64byte).

nodeInfos オブジェクト

11.3.6. inits オブジェクト

inits オブジェクトは、SW-4230-1000 CO2 ノードのように動作パラメータを設定できるノードに対し、PD Handler SW4x 起動時にノードへ送信する設定コマンドを指定するためのオブジェクトです。

キー	データ型	説明
enable	論理型	この初期化処理を行うか否か.
rid	整数又は文字列	対象ノードの RIDを '0xNN' 形式の16進数もしくは整数で指定する.
cmd	文字列	'0x' から始まる 26 文字のコマンド文字列コマンド文字列.
description	文字列	モデリングファイル上のコメント.

inits オブジェクト

11.3.7. polls オブジェクト

polls オブジェクトは、SW-4230-1000 CO2 ノードのように任意にデータを読み出せるノードに対する、ポーリング処理の諸元を指定します。
SW-4280-1000 Modbus RTU ノード用とその他のモデル用(汎用)で、指定可能なオブジェクトが異ります。

11.3.7.1. 汎用

キー	データ型	説明
enable	論理型	このポーリング処理を行うか否か.
rid	整数又は文字列	対象ノードの RIDを '0xNN' 形式の16進数もしくは整数で指定する.
cmd	文字列	'0x' から始まる 26 文字のコマンド文字列コマンド文字列.
interval	整数値	データを取得する間隔(sec). デフォルト値は 60.
time_sync	論理値	基準時刻制御モード. デフォルト値は false.
base_time	文字列	基準時刻制御モードの基準時刻を 'HH:MM' 形式で指定する. デフォルト値は '00:00'.
description	文字列	モデリングファイル上のコメント.

汎用 pools オブジェクト

11.3.7.2. SW-4280-1000 Modbus RTU ノード用

キー	データ型	説明
enable	論理型	このポーリング処理を行うか否か.
rid	整数又は文字列	対象ノードの RIDを '0xNN' 形式の16進数もしくは整数で指定する.
unitId	整数又は文字列	PLC機器の Modbus ユニット ID. 1 ～ 247 又は 255 (TCPプロトコルのみ). '0x' で始まる 16進数表記の文字列入力も可能.
maker	文字列	Modbus モデルファイルに定義されるメーカー名 (64byte).
product	文字列	Modbus モデルファイルに定義されるプロダクト名 (64byte).
model	文字列	Modbus モデルファイルに定義されるmodel名 (64byte).
sku	文字列	Modbus モデルファイルに定義される sku 名 (64byte).
interval	整数値	データを取得する間隔(sec). デフォルト値は 60.
time_sync	論理値	基準時刻制御モード. デフォルト値は false.
base_time	文字列	基準時刻制御モードの基準時刻を 'HH:MM' 形式で指定する. デフォルト値は '00:00'.
description	文字列	モデリングファイル上のコメント.

SW-4280-1000 Modbus RTU ノード用 pools オブジェクト

11.3.8. モデル別オブジェクト

11.3.8.1. SW4X_COMMON オブジェクト

SW4X_COMMON オブジェクトには、各モデル共通のキー等が設定されます。

SW4X_COMMON オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
keys	JSON配列	オブジェクト数 25
infos	JSON obj	
nodeInfos	JSON obj	

SW4X_COMMON オブジェクトで設定可能なブジェクト

SW4X_COMMON オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	timestamp	true			ISO8601 拡張書式タイムスタンプ
2	unitId	true	HEX		SID データの送信元 ID
3	unitType	true	HEX		ユニットのタイプコード
4	model	false			モデル名
5	description	false			モデルの機能概要
6	ctrlCode	false	HEX		制御コード
7	batteryVoltage	true	HEX		電池電圧の状態コード
8	swVersion	true			ノードのソフトウェアバージョン NNN.NNN.NNN
9	routes	true			経路情報を付加するか否か
10	unitId		HEX		経路情報のデータ中継ユニット ID
11	RSSI		HEX		経路情報の RSSI
12	rid	true	HEX		RID
13	gid	false	HEX		GID
14	rfChannel	false	dec		無線チャンネル番号
15	rfPower	false	dec		無線出力 1 or 20 (mW)
16	cmd	true			コマンド文字列
17	msg	true			メッセージ文字列
18	nodeInfos	false			nodeInfos オブジェクト
19	errorCode	true	HEX		エラーコード
20	rawData				RAW モード時のデータ
21	request_from	true			応答メッセージに含まれるリクエスト元クラウド ID
22	reply_to	true			応答メッセージに含まれるリクエストメッセージの MD5
23	result	true	bool		応答メッセージに含まれるリクエスト実行の成否
24	reason	true			応答メッセージに含まれるリクエスト失敗時のエラーメッセージ
25	request	true			応答メッセージに含まれるリクエストメッセージ

SW4X_COMMON の keys オブジェクト

11.3.8.2. SW42P0_x01 オブジェクト

SW42P0_1x01 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 2
infos	JSON obj	
precision	整数値	デフォルト値 5

SW42P0_1x01 オブジェクト

SW42P0_1x01 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	temperature	true		-1	温度(℃)　-79.9 ～ 79.9
2	tempeStaus	true	bool		温度の測定ステータス

SW42P0_1x01 の keys オブジェクト

11.3.8.3. SW4210_1202 オブジェクト

SW4210_1202 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 4
infos	JSON obj	
precision	整数値	デフォルト値 5

SW4210_1202 オブジェクト

SW4210_1202 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	temperature	true		-1	温度(℃)　-79.9 ～ 79.9
2	humidity	true		-1	湿度(%)　0.0 ～ 100.0
3	tempeStaus	true	bool		温度の測定ステータス
4	humidStaus	true	bool		湿度の測定ステータス

SW4210_1202 の keys オブジェクト

11.3.8.4. SW4210_1205 オブジェクト

SW4210_1205 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 2
infos	JSON obj	
precision	整数値	デフォルト値 5

SW4210_1205 オブジェクト

SW4210_1205オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	illuminance	true		0	照度(lx)　0 ～ 99999
2	illumStatus	true	bool		照度の測定ステータス

SW4210_1205 の keys オブジェクト

11.3.8.5. SW4210_1204 オブジェクト

SW4210_1204 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 6
infos	JSON obj	
precision	整数値	デフォルト値 5

SW4210_1204 オブジェクト

SW4210_1204オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	temperature	true		-1	温度(℃)　-79.9 ～ 79.9
2	humidity	true		-1	湿度(%)　0.0 ～ 100.0
3	illuminance	true		0	照度(lx)　0 ～ 99999
4	tempeStaus	true	bool		温度の測定ステータス
5	humidStaus	true	bool		湿度の測定ステータス
6	illumStatus	true	bool		照度の測定ステータス

SW4210_1204 の keys オブジェクト

11.3.8.6. SW4220_1000 オブジェクト

SW4220_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 3
infos	JSON obj	
precision	整数値	デフォルト値 17

SW4220_1000 オブジェクト

SW4220_1000オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	humanCount	true		0	人感検出回数　0 ～ 4095
2	detectionMaxWidth	true		1	人感検出幅 NAX　0 ～ 255
3	detectionMinWidth	true		1	人感検出幅 MIN　0 ～ 255

SW4220_1000 の keys オブジェクト

11.3.8.7. SW4240_1000 オブジェクト

SW4240_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 3
infos	JSON obj	
precision	整数値	デフォルト値 17

SW4240_1000 オブジェクト

SW4240_1000オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	pulseCount1	true		0	パルスカウント数１　0 ～ 99999999
2	pulseCount2	true		0	パルスカウント数２　0 ～ 99999999
3	eepromStatus	true	bool		EEPROM ステータス

SW4240_1000 の keys オブジェクト

11.3.8.8. SW4220_1020 オブジェクト

SW4220_1020 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 2
infos	JSON obj	
precision	整数値	デフォルト値 17

SW4220_1020 オブジェクト

SW4220_1020 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	humanDetectCount	true		0	人感検知カウント　0 ～ 99999999
2	periodic	true	bool		定期送信を示すフラグ

SW4220_1020 の keys オブジェクト

11.3.8.9. SW42J0_1202 オブジェクト

SW42J0_1202 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 7
infos	JSON obj	
precision	整数値	デフォルト値 5

SW42J0_1202 オブジェクト

SW42J0_1202 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	temperature	true		-1	温度(℃)　-79.9 ～ 79.9
2	humidity	true		-1	湿度(%)　0.0 ～ 100.0
3	tempeStaus	true	bool		温度の測定ステータス
4	humidStaus	true	bool		湿度の測定ステータス
5	subCrtlCode	true	HEX		サブ制御コード
6	confData	true			設定データ．ノードの設定情報バイト列.
7	cmdRespons	true			受信確認レスポンスバイト列.

SW42J0_1202 の keys オブジェクト

11.3.8.10. SW42K0_1000 オブジェクト

SW42K0_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 2
infos	JSON obj	
precision	整数値	デフォルト値 15

SW42K0_1000 オブジェクト

SW42K0_1000 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	integralPowerConsum	true		-4	積算電力(Kwh)　0.0 ～ 99999999.9999
2	eepromStatus	true	bool		EEPROM ステータス

SW42K0_1000 の keys オブジェクト

11.3.8.11. SW42DD_1000 オブジェクト

SW42DD_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 26
infos	JSON obj	
precision	整数値	デフォルト値 15

SW42DD_1000 オブジェクト

SW42DD_1000 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	distributionSystem12	true			CH1,2 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
2	distributionSystem34	true			CH3,4 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
3	outputChannel1	true	bool		CH1 の出力状態
4	outputChannel2	true	bool		CH2 の出力状態
5	outputChannel3	true	bool		CH3 の出力状態
6	outputChannel4	true	bool		CH4 の出力状態
7	meanCurrent1	true		-3	CH1 平均電流(A)　0.0 ～ 9.999
8	meanCurrent2	true		-3	CH2 平均電流(A)　0.0 ～ 9.999
9	meanCurrent3	true		-3	CH3 平均電流(A)　0.0 ～ 9.999
10	meanCurrent4	true		-3	CH4 平均電流(A)　0.0 ～ 9.999
11	instantCurrent1	true		-3	CH1 瞬時電流(A)　0.0 ～ 9.999
12	instantCurrent2	true		-3	CH2 瞬時電流(A)　0.0 ～ 9.999
13	instantCurrent3	true		-3	CH3 瞬時電流(A)　0.0 ～ 9.999
14	instantCurrent4	true		-3	CH4 瞬時電流(A)　0.0 ～ 9.999
15	meanPower1	true		-3	CH1 平均電力(Kw)　0.0 ～ 9.999
16	meanPower2	true		-3	CH2 平均電力(Kw)　0.0 ～ 9.999
17	meanPower3	true		-3	CH3 平均電力(Kw)　0.0 ～ 9.999
18	meanPower4	true		-3	CH4 平均電力(Kw)　0.0 ～ 9.999
19	instantPower1	true		-3	CH1 瞬時電力(Kw)　0.0 ～ 9.999
20	instantPower2	true		-3	CH2 瞬時電力(Kw)　0.0 ～ 9.999
21	instantPower3	true		-3	CH3 瞬時電力(Kw)　0.0 ～ 9.999
22	instantPower4	true		-3	CH4 瞬時電力(Kw)　0.0 ～ 9.999
23	integralPowerConsum1	true		-2	CH1 積算有効電力量(Kwh) 0.0 ～ 9999999.99
24	integralPowerConsum2	true		-2	CH2 積算有効電力量(Kwh) 0.0 ～ 9999999.99
25	integralPowerConsum3	true		-2	CH3 積算有効電力量(Kwh) 0.0 ～ 9999999.99
26	integralPowerConsum4	true		-2	CH4 積算有効電力量(Kwh) 0.0 ～ 9999999.99

SW42DD_1000 の keys オブジェクト

11.3.8.12. SW42DD_1100 オブジェクト

SW42DD_1100 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 26
infos	JSON obj	
precision	整数値	デフォルト値 15

SW42DD_1100 オブジェクト

SW42DD_1100 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	distributionSystem12	true			CH1,2 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
2	distributionSystem34	true			CH3,4 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
3	outputChannel1	true	bool		CH1 の出力状態
4	outputChannel2	true	bool		CH2 の出力状態
5	outputChannel3	true	bool		CH3 の出力状態
6	outputChannel4	true	bool		CH4 の出力状態
7	meanCurrent1	true		-1	CH1 平均電流(A)　0.0 ～ 999.9
8	meanCurrent2	true		-1	CH2 平均電流(A)　0.0 ～ 999.9
9	meanCurrent3	true		-1	CH3 平均電流(A)　0.0 ～ 999.9
10	meanCurrent4	true		-1	CH4 平均電流(A)　0.0 ～ 999.9
11	instantCurrent1	true		-1	CH1 瞬時電流(A)　0.0 ～ 999.9
12	instantCurrent2	true		-1	CH2 瞬時電流(A)　0.0 ～ 999.9
13	instantCurrent3	true		-1	CH3 瞬時電流(A)　0.0 ～ 999.9
14	instantCurrent4	true		-1	CH4 瞬時電流(A)　0.0 ～ 999.9
15	meanPower1	true		-2	CH1 平均電力(Kw)　0.0 ～ 99.99
16	meanPower2	true		-2	CH2 平均電力(Kw)　0.0 ～ 99.99
17	meanPower3	true		-2	CH3 平均電力(Kw)　0.0 ～ 99.99
18	meanPower4	true		-2	CH4 平均電力(Kw)　0.0 ～ 99.99
19	instantPower1	true		-2	CH1 瞬時電力(Kw)　0.0 ～ 99.99
20	instantPower2	true		-2	CH2 瞬時電力(Kw)　0.0 ～ 99.99
21	instantPower3	true		-2	CH3 瞬時電力(Kw)　0.0 ～ 99.99
22	instantPower4	true		-2	CH4 瞬時電力(Kw)　0.0 ～ 99.99
23	integralPowerConsum1	true		-2	CH1 積算有効電力量(Kwh) 0.0 ～ 9999999.99
24	integralPowerConsum2	true		-2	CH2 積算有効電力量(Kwh) 0.0 ～ 9999999.99
25	integralPowerConsum3	true		-2	CH3 積算有効電力量(Kwh) 0.0 ～ 9999999.99
26	integralPowerConsum4	true		-2	CH4 積算有効電力量(Kwh) 0.0 ～ 9999999.99

SW42DD_1100 の keys オブジェクト

11.3.8.13. SW42D0_1000 オブジェクト

SW42D0_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 19
infos	JSON obj	
precision	整数値	デフォルト値 15

SW42D0_1000 オブジェクト

SW42D0_1000 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	channel1LowCurrent	true	bool		CH1 の電流が 1A 未満
2	distributionSystem12	true			CH1,2 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
3	distributionSystem34	true			CH3,4 の配電方式. '1P2W(1CT)', '3P3W(1CT)', '1P3W(2CT)', '3P3W(2CT)' のいずれか.
4	outputChannel1	true	bool		CH1 の出力状態
5	outputChannel2	true	bool		CH2 の出力状態
6	outputChannel3	true	bool		CH3 の出力状態
7	outputChannel4	true	bool		CH4 の出力状態
8	current1	true		-1	CH1 電流(A)　0.0 ～ 999.9
9	current2	true		-1	CH2 電流(A)　0.0 ～ 999.9
10	current3	true		-1	CH3 電流(A)　0.0 ～ 999.9
11	current4	true		-1	CH4 電流(A)　0.0 ～ 999.9
12	power1	true		-1	CH1 電力(Kw)　0.0 ～ 999.9
13	power2	true		-1	CH2 電力(Kw)　0.0 ～ 999.9
14	power3	true		-1	CH3 電力(Kw)　0.0 ～ 999.9
15	power4	true		-1	CH4 電力(Kw)　0.0 ～ 999.9
16	integralPowerConsum1	true		-1	CH1 積算有効電力量(Kwh) 0.0 ～ 9999999.99
17	integralPowerConsum2	true		-1	CH2 積算有効電力量(Kwh) 0.0 ～ 99999999.9
18	integralPowerConsum3	true		-1	CH3 積算有効電力量(Kwh) 0.0 ～ 99999999.9
19	integralPowerConsum4	true		-1	CH4 積算有効電力量(Kwh) 0.0 ～ 99999999.9

SW42D0_1000 の keys オブジェクト

11.3.8.14. SW4230_1000 オブジェクト

SW4230_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 3
inits	JSON配列	
polls	JSON配列	汎用 polls オブジェクト
infos	JSON obj	
precision	整数値	デフォルト値 17

SW4230_1000 オブジェクト

SW4230_1000オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	co2	true		0	二酸化炭素濃度（ppm） 0 ～ 99999
2	altitude	true		0	高度（m） 0 ～ 9999
3	periodicCode	true	HEX		読みだし周期コード. '0x00:無し', '0x01:1分', '0x02:5分', '0x03:10分' のいずれか

SW4230_1000 の keys オブジェクト

11.3.8.15. SW4260_1x10 オブジェクト

SW4260_1x10 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 15
inits	JSON配列	
polls	JSON配列	汎用 polls オブジェクト
infos	JSON obj	
precision	整数値	デフォルト値 15

SW4260_1x10 オブジェクト

SW4260_1x10 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	sensorNo	true	dec		センサー番号
2	voltageP1P2	true		-2	P1-P2 間電圧（V） 0.00 ～ 99999.90
3	voltageP2P3	true		-2	P2-P3 間電圧（V） 0.00 ～ 99999.90
4	current1	true		-2	I1 電流（A） 0.00 ～ 9999.99
5	current2	true		-2	I2 電流（A） 0.00 ～ 9999.99
6	activePower	true		-2	有効電力（Kw） -9999999.99 ～ 9999999.99
7	powerFactor	true		-2	力率（%） -1.00 ～ 1.00
8	frequency	true		-2	周波数（Hz） 45.00 ～ 65.00
9	integralPowerConsum	true		-2	積算電力量（kwh） 0.00 ～ 999999999.00
10	sensorStatus	true			センサーステータス 0xNNNN
11	sensorVersion	true			センサースバージョン 0xNNNNNNNN
12	sensorParameter	true	dec		センサーパラメタ番号 0 ～ 9
13	sensorConfig	true			センサーパラメタ設定値 0xNNNNNN
14	periodicCode	true	HEX		読みだし周期コード. '0x00:無し', '0x01:1分', '0x02:5 分', '0x03:10 分' のいずれか
15	sensorConnected	true			接続センサー番号 0xNNNNNNNN

SW4260_1x10 の keys オブジェクト

11.3.8.16. SW4280_1000 オブジェクト

SW4280_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 4
inits	JSON配列	
polls	JSON配列	SW-4280-1000 Modbus RTU ノード用 polls オブジェクト
infos	JSON obj	
precision	整数値	Modbus モデリングファイルの設定を優先.

SW4280_1000 オブジェクト

SW4280_1000オブジェクトに記載される keysオブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	length	true	dec		Modbus メッセージ長 6 ～ 10
2	modbusId	true	HEX		Modbus アドレス. Modbus のユニット ID 0xNN
3	unitId	true	HEX		SID データの送信元 ID
4	modbusFunction	true	HEX		Modbus ファンクション. Modbus のファンクション番号 0xNN
5	modbusData	true			Modbus ノードを介して取得したデータ 0xNNNNNNNNNNNNNNNN

SW4280_1000 の keys オブジェクト

polls オブジェクトに設定されたポーリング処理は Modbus モデリングファイルに応じたキー/バリューフォーマットで出力されます。

本 keys オブジェクトに設定されるフォームは、下流方向制御等によって SW-4280-1000 Modbus RTU ノードより意図しないメッセージが、送られて来た場合にのみ適用されます。

11.3.8.17. SW4260_1x20 オブジェクト

SW4260_1x20 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 18
inits	JSON配列	
polls	JSON配列	汎用 polls オブジェクト
infos	JSON obj	
precision	整数値	デフォルト値 15

SW4260_1x20 オブジェクト

SW4260_1x20 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	circuitNo	true	dec		回線番号
2	voltage1	true		-2	電圧 1 (V)　0.00 ～ 999999.90
3	voltage2	true		-2	電圧 2 (V)　0.00 ～ 999999.90
4	voltage3	true		-2	電圧 3 (V)　0.00 ～ 999999.90
5	current1	true		-2	電流 1 (A)　0.00 ～　99999.99
6	current2	true		-2	電流 2 (A)　0.00 ～　99999.99
7	current3	true		-2	電流 3 (A)　0.00 ～　99999.99
8	activePower	true		-2	有効電力 (Kw) -214748.36 ～ 214748.36
9	activePowerW	true		-2	有効電力 (W) -214748364.70 ～ 214748364.70
10	reactivePower	true		-2	無効電力 (Var) -214748364.70 ～ 214748364.70
11	powerFactor	true		-2	力率 (%)　-1.00 ～ 1.00
12	frequency	true		-2	周波数 (Hz)　45.00 ～ 65.00
13	integralPowerConsum	true		-2	積算電力量 (Kwh) 0.00 ～ 999999999.00
14	integralPowerConsumW	true		-2	積算電力量 (Wh) 0.00 ～ 999999999.00
15	sensorStatus	true			センサーステータス 0xNNNN
16	sensorVersion	true			センサースバージョン 0xNNNNNNNN
17	periodicCode	true	HEX		読みだし周期コード. '0x00:無し', '0x01:1 分', '0x02:5 分', '0x03:10 分' のいずれか
18	sensorConnected	true			接続回線番号 0xNNNNNNNN

SW4260_1x20 の keys オブジェクト

11.3.8.18. SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクト

SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクトで設定可能なブジェクトを示します。

キー	データ型	説明
enable	論理型	デフォルト値 true
keys	JSON配列	オブジェクト数 1
infos	JSON obj	

SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクト

SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 オブジェクトに記載される keys オブジェクトの並びとデフォルト値を示します。

順列	デフォルト値				
	key	include	form	ratio	
1	periodic	true	bool		定期送信を示すフラグ

SW4000_1000, SW4100_1000, SB4020_1000, SW4300_1000, SW4500_1000 の keys オブジェクト

11.4. RTU モデリングファイルの書式

RTU モデリングファイルは、モデリングの内 SW4280_1000 オブジェクトの inits オブジェクトと polls オブジェクトを別ファイルで設定するために用意されたファイルです。PD Handler SW4x の RTU モデリングファイルは次の書式で構成されます。

```
{
    inits オブジェクトの配列,
    SW-4280-1000 Modbus RTU ノード用の polls オブジェクトの配列
}
```

inits オブジェクトと SW-4280-1000 Modbus RTU ノード用の polls オブジェクトの書式は、モデリングファイルと同じです。

11.5. 基準時刻制御モード

基準時刻制御は、polls オブジェクトの time_sync で有効となり、同オブジェクトの interval と base_time の設定に基づき、定時刻にデータの収集を行う機能です。
基準時刻制御における「取得時間間隔」は 60, 120, 180, 300, 600, 900, 1800, 3600, 7200, 10800, 14400, 21600, 28800, 43200 と 86400 の倍数に限られます。「取得時間間隔」としてこれら以外の値が設定されると PD Handler Modbus 2 Client 内で次のように扱われます。

取得時間間隔の設定値	実動作値
0 ～ 119	60
120 ～ 179	120
180 ～ 299	180
300 ～ 599	300
600 ～ 899	600
900 ～ 1799	900
1800 ～ 3599	1800
3600 ～ 7199	3600
7200 ～ 10799	7200
10800 ～ 14399	10800
14400 ～ 21599	14400
21600 ～ 28799	21600
28800 ～ 43199	28800
43200 ～ 86399	43200
86400～	86400 の倍数

基準時刻制御における取得時間間隔の設定と実動作値

「基準時刻」とは動作の起点となる時刻で、例えば「取得時間間隔」を 300 とし、「基準時刻」を”00:01”とした場合、データの取得は 00:01, 00:06, 00:11 ... 00:56, 01:01 ... 23:56, 00:01 の定刻に行われます。
データの取得開始時刻は「基準時刻」に設定さた時刻そのものではなく、「基準時刻」と「取得時間間隔」から算定される直近の時刻となります。
例えば 08:30 に「基準時刻」”01:05”、 「取得時間間隔」10800 の設定が行われた場合、最初のデータ取得は 10:05 に行われ、以降 13:05, 16:05, 19:05, 22:05, 01:05 の順におこなわれます。

11.6. Modbus モデリングファイルの書式

PD Handler SW4x の Modbus モデリングファイルはの書式は、PD Handler Modbus 2 のモデリングファイルと同じです。
デフォルトでは PD Handler Modbus 2 と同じファイルを用います。

11.7. 設定ファイルの書式

11.7.1. 構文

PD Handler sw4x の設定ファイルは次の書式で構成されています。

```
{
    "custom_file": "カスタマイズ用設定ファイル",
    "sw4x_model_file": "モデリングファイル",
    "sw4x_rtu_file": "RTU モデリングファイル",
    "model_file": "Modbus モデリングファイル",
    "localname": "デバイスのローカル名",
    "enable": 論理値,
    "bind": "データを受け取るソケット名",
    "push_to": "データの送り先ソケット名",
    "buffer_size": データのバッファサイズ,
    "receive": 論理値,
    "reply": 論理値,
    "raw_mode": 論理値,
    "uart_device": "シリアル接続デバイス名",
    "uart_speed": シリアル接続ビットレート,
    "uart_bits": シリアル接続ビット数,
    "uart_stop": シリアル接続のストップビット,
    "uart_parity": "シリアル接続のパリティ",
    "gid": 無線ネットワークの GID,
    "power20": 論理値,
    "channel": 無線チャンネル,
    "rtt": Modbus RTU ノードの Round trip time,
    "infos" : "出力に付加されるユーザー定義オブジェクト."
}
```

11.7.2. 設定オブジェクト

キー	データ型	説明
custom_file	文字列	カスタマイズ設定用ファイル. デフォルト値は、'/var/webui/config/pd_handler_sw4x_custom.conf'. (MAXPATHLEN)
sw4x_model_file	文字列	モデリングファイル . デフォルト値は'/var/webui/config/pd_handler_sw4x_model.conf'. ファイルが実在しない場合は、ハードコーディングされたモデリングが用いられる. (MAXPATHLEN)
sw4x_rtu	文字列	RTU モデリングファイル . デフォルト値は 、'/var/webui/config/pd_handler_sw4x_rtu.conf'.
model_file	文字列	Modbusモデリングファイル. デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
enable	論理値	デフォルト値はfalse.
localname	文字列	デバイスのローカル名(デバイス番号). (32byte)
bind	文字列	データを受け取るソケット名. 文字列の先頭が'@' の場合はabstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_handler/<デバイス番号>.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名. 文字列の先頭が'@' の場合は abstract namespace と解釈します. 空の場合は、デフォルト値 @/pd_repeater/<デバイス番号>.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte). デフォルト値は4096
receive	論理値	クラウドからメッセージを受け取る下流方向制御) か否か. デフォルト値はfalse.
reply	論理値	下流方向制御に対しステータスメッセージを返すか否か. デフォルト値は false.
raw	論理値	データをパースせずにそのまま文字列として送信する. デフォルト値は false.
uart_device	文字列	シリアル接続のデバイス名. デフォルト値は '/dev/ttyEX2'
gid	整数値	無線ネットワークの GID. 101 ～ 254. デフォルト値は 101.
power20	論理値	無線出力を 1mW から 20mW に切替えるか否か. デフォルト値は true.
channel	整数値	無線チャンネル. 無線出力 1mW 時: 25 ～ 75、20mW時: 25～60、ただし 32,33,61,62,63 を除く. デフォルト値は 60.
rtt	整数値	Modbus RTU ノードの Round trip time(msec). デフォルト値は 500(ms).
infos	JSON obj	出力に付加されるユーザー定義オブジェクト.

PD Handler sw4x の設定 オブジェクト

11.8. カスタマイズ用設定ファイルの書式

カスタマイズ用設定ファイルは、Web UI の出力する設定ファイルに対し、sw4x_model_file , sw4x_rtu_file , model_file ,bind, push_to, buffer_size,infos の各値をカスタマイズ(オーバーライト)します。

11.8.1. 構文

PD Handler sw4x の設定ファイルは次の書式で構成されています。

```
{
    "sw4x_model_file": "モデリングファイル",
    "sw4x_rtu_file": "RTU モデリングファイル",
    "model_file": "Modbus モデリングファイル",
    "bind": "データを受け取るソケット名",
    "push_to": "データの送り先ソケット名",
    "buffer_size": データのバッファサイズ,
    "infos" : "出力に付加されるユーザー定義オブジェクト."
}
```

11.8.2. カスタマイズ用設定オブジェクト

キー	データ型	説明
sw4x_model_file	文字列	モデリングファイル．デフォルト値は '/var/webui/config/pd_handler_sw4x_model.conf'. ファイルが実在しない場合は、ハードコーディングされたモデリングが用いられる．(MAXPATHLEN)
sw4x_rtu	文字列	RTU モデリングファイル．デフォルト値は 、'/var/webui/config/pd_handler_sw4x_rtu.conf'.
model_file	文字列	Modbusモデリングファイル．デフォルト値は、'/var/webui/config/pd_handler_modbus_2_model.conf'. (MAXPATHLEN)
bind	文字列	データを受け取るソケット名．文字列の先頭が'@' の場合はabstract namespace と解釈します． 空の場合は、デフォルト値 @/pd_handler/＜デバイス番号＞.sock が設定されます.
push_to	文字列	制御メッセージの送り先ソケット名．文字列の先頭が'@' の場合は abstract namespace と解釈します． 空の場合は、デフォルト値 @/pd_repeater/＜デバイス番号＞.sock が設定されます.
buffer_size	整数値	データのバッファサイズ(byte)．デフォルト値は4096
infos	JSON obj	出力に付加されるユーザー定義オブジェクト.

PD Handler sw4x のカスタマイズ用設定 オブジェクト

11.9. メッセージオブジェクト

11.9.1. コマンド入力オブジェクト

コマンド入力に用いられる JSON オブジェクトを示します。

キー	データ型	説明
rid	文字列	制御対象ノードの RID を '0xNN' 形式で指定する.
cmd	文字列	制御対象ノードに対する CMD のバイト列を '0xNNNNNNNNNNNNNNNNNNNNNNNN' 形式で指定する. 文字列が 2byte + 24byte に満たない場合、下位のバイトは '0x00' で埋めれる. 以下の bNN, unitType, ctlCd, circuitNumber が設定された場合、上書される.
bNN	文字列	CMD バイト列の 'NN' 番目に設定するバイト列を '0xNNnnnn' 形式で指定する. 'NNnnnn' は任意長のバイト列. 例えば 'B07 - B04' を '0x12345678' とする場合は '"b07":"0x12345678"' と指定する.
unitType	文字列	ユニットタイプを '0xNN' 形式で指定する. '"b11":"0xNN"' と同義。
ctlCd	文字列	制御コードを '0xNN' 形式で指定する. '"b10":"0xNN"' と同義.
subCtlCd	文字列	サブ制御コードを '0xNN' 形式で指定する. '"b09":"0xNN"' と同義.
sensorNumber	文字列	センサー番号を '0xNN' 形式で指定する. '"b08":"0xNN"' と同義.
circuitNumber	文字列	回路番号を '0xNN' 形式で指定する. '"b08":"0xNN"' と同義

コマンド入力オブジェクト

11.9.2. 応答メッセージオブジェクト

SW4000 シリーズ無線センサーネットワークは、コマンド入力に対し非同期に応答するため、応答メッセージは入力されたコマンドが無線センサーネットワークへの出力されたことを通知するものであり、各ノードにおけるコマンドの実行結果については通常のデータ出力メッセージとしてハンドリングされます。
応答メッセージは設定オブジェクトの reply キーが true に設定されている場合に出力されます。
応答メッセージに用いられる JSON オブジェクトをしめします。

キー	内容	データ型	備考
timestamp	コマンド出力時刻	文字列	ISO8601 拡張書式
request_from	リクエストメッセージの送付元クラウド ID	文字列	
reply_to	リクエストメッセージの MD5	文字列	
result	出力ステータス	論理値	
reason	エラーの理由	文字列	
request	入力されたコマンド	文字列	
gid	GID	文字列	
rfChannel	無線チャンネル番号	整数値	
rfPower	無線出力	整数値	
nodeInfos	nodeInfos オブジェクト	JSON obj	
infos	infos オブジェクト	JSON obj	

応答メッセージオブジェクト

キーの名称と一部のデータ型は、SW4X_COMMN オブジェクトの設定により変更することができます。

11.9.3. クラウドからの制御

例えば、RID’0xFA’の SW4230¥_1000 CO2 ノードの標高設定を 1234m に設定するのであれば、次のメッセージを送ります。

```
{
    "rid":"0xFA",
    "cmd":"0x2001020000000000000001234"
}
```

設定オブジェクトの reply キーが true に設定されている場合、次の応答メッセージが返されます。

```
{
    "timestamp": "2019-11-14T11:30:08.949+09:00"
    "request_from":"0x13(0)",
    "reply_to":"1bf33e22f0c27bcbc9a933580da6bb72",
    "result": true
}
```

11.9.4. CLI(pd_handler_sw4x_cli)

pd_handler_sw4x_cli は、クラウドからの制御をローカルに行うためのアプリケーションです。
pd_handler_sw4x とはインタフェースが競合するため、pd_handler_sw4x を停止した上で使用して下さい。

pd_handler_sw4x_cli のオプションは次の通りです。

```
root# /usr/sbin/pd_handler_sw4x_cli -h
usages: pd_handler_sw4x_cli [-dqs][-c <conf file>] <RID> <CMD string>
  -c conf file     Conf file. [default: /var/webui/config/pd_handler_sw4x.conf]
  -d               Debug mode.
  -q               Quiet mode.
  -s               Write property to module.
```

<RID>は、制御対象ノードの RID を '0xNN' 形式で指定します。
<CMD string>は、制御対象ノードに対する CMD のバイト列を
'0xNNNNNNNNNNNNNNNNNNNNNNNN' 形式で指定します。
文字列が 2byte + 24byte に満たない場合、下位のバイトは '0x00' で埋めれます。
'-s' オプションは SB-4020 通信モジュールに対する書き込みを行うためのオプションです。

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